Scraping X

Francesco Pinsone

Indice

1 Introduzione

L'obiettivo è quello di estrarre dati sensibili su possibili minacce informatiche da X. La suite di script implementati si propone quindi di effettuare un'attività di scraping su X utilizzando Le librerie Python `Selenium` e `BeautifulSoup` . Il software riceve input diversi a seconda di quale script viene utilizzato e da quale specifica attività di scraping si vuole svolgere. Nel caso di vogliano estrarre tweet per parole chiave si passa in input una lista di parole chiave e una lista target di gruppi hacker particolarmente attivi in Europa e nel mondo, sulla base delle quali verrà fatta la ricerca. Se si vogliono estrarre le informazioni degli utenti il software estrapola la lista degli utenti che hanno pubblicato i tweet fino ad ora estratti e salvati su MongoDB e per ognuno di essi effettua lo scraping delle informazioni. Se si vogliono invece ottenere gli utenti correlati il codice ricava una lista di utenti target dagli utenti di cui abbiamo precedentemente salvato le informazioni su MongoDB e per ognuno di essi ricava gli utenti correlati e ne estrae le informazioni.
In uscita si ottiene invece che i dati estratti vengono organizzati in oggetti JSON che vengono poi salvati su un apposito database MongoDB.

2 Selenium & BeautifulSoup

Selenium e BeautifulSoup sono due librerie di Python comunemente utilizzate per il *web scraping*, ovvero l'estrazione automatizzata di dati da pagine web. Selenium, una libreria versatile e potente, consente di controllare un browser web in modo programmatico, simulando azioni dell'utente come il caricamento delle pagine, i click su bottoni e il riempimento di form. Questo approccio è particolarmente utile per gestire contenuti dinamici o interattivi, come quelli generati da JavaScript, che altrimenti non sarebbero accessibili tramite semplici richieste HTTP. BeautifulSoup, d'altra parte, è una libreria di parsing per HTML e XML che permette di analizzare e manipolare il codice sorgente delle pagine, facilitando l'identificazione e l'estrazione dei dati desiderati tramite una sintassi intuitiva.
Selenium viene utilizzato per interagire con le pagine web, scorrere tra i contenuti e attendere il caricamento di sezioni specifiche. Una volta che la pagina è completamente caricata, *BeautifulSoup* entra in gioco per analizzare il codice HTML e isolare i dati rilevanti (come testo, immagini o altri elementi), strutturandoli in un formato più facilmente elaborabile. Questa combinazione rende i due strumenti particolarmente efficaci per raccogliere dati da siti complessi, permettendo una navigazione automatica del sito tramite *Selenium* e un'efficace estrazione delle informazioni con *BeautifulSoup*.

3 Scraping

Il codice implementato si suddivide in più gruppi di script Python, preposti allo svolgimento di compiti diversi, seppur con delle caratteristiche comuni.

Dato che Selenium consente di automatizzare diverse tipologie di browser il codice è stato sviluppato nel medesimo modo in due differenti versioni: una che utilizza Google Chrome e una che utilizza Firefox.

Per una descrizione più approfondita del codice sorgente si rimanda alla documentazione dettagliata.

3.1 Scraping dei Tweets

Questo script è stato progettato per automatizzare l'estrazione di informazioni da X (ex Twitter), con lo scopo di analizzare tweet relativi a gruppi hacker specifici menzionati in un report sulla cybersecurity del 2024. Lo script utilizza librerie per l'automazione del browser e il parsing HTML come `Selenium` e `BeautifulSoup`, rispettivamente. Per avviare la procedura, il programma carica una lista di gruppi target da un database `MongoDB`, e, tramite credenziali precedentemente salvate, effettua il login a X per accedere alle funzionalità di ricerca della piattaforma.
Per ciascun gruppo target, lo script effettua ricerche su X, costruendo l'URL con parole chiave casuali per ogni gruppo e completando così una ricerca mirata in lingua inglese, senza link né risposte. Una volta caricata la pagina, scorrono automaticamente verso il basso per caricare un numero maggiore di tweet, quindi acquisisce la struttura HTML della pagina. Utilizzando `BeautifulSoup`, analizza il contenuto HTML per identificare i `div` contenenti i tweet rilevanti. Da ogni tweet estrae vari dati, tra cui testo, URL, immagini, e video.
Le informazioni raccolte vengono salvate nel database per consentire analisi future. In modo simile, un modulo dedicato permette di estrarre e salvare informazioni di profilo utente rilevanti (come nome utente, verifica, follower, e dettagli di contatto) per ciascun profilo menzionato nei tweet.

3.2 Scraping delle Informazioni Utente

Per l'estrazione automatica delle informazioni di un utente X, è stato sviluppato uno gruppo di script che automatizza il processo, dall'identificazione degli utenti alla registrazione dei dati in un database. L'idea alla base è di costruire un sistema che recuperi periodicamente informazioni sugli utenti da analizzare, partendo da tweet raccolti in precedenza e salvati in un database.
Il processo si avvia con la connessione al database, da cui vengono selezionati gli utenti da analizzare, escludendo i già processati. Lo script esegue poi l'accesso a X mediante `Selenium`. Per ogni utente, Selenium naviga sul profilo di X, raccoglie il contenuto della pagina e verifica se l'utente è effettivamente esistente o se ha accesso limitato, bypassando eventuali blocchi temporanei.
Una volta che il contenuto della pagina è stato caricato, lo script sfrutta `BeautifulSoup`, per estrarre le informazioni chiave. Queste includono dettagli sull'utente come nome, stato verificato, numero di post e follower, lavoro, località, sito web e altre informazioni biografiche, se presenti. BeautifulSoup opera individuando

elementi specifici attraverso tag HTML, recuperando informazioni visibili e testuali o verificando la presenza di immagini, video e link.
Le informazioni raccolte sono infine salvate in una collezione dedicata del database, per poterle utilizzare nelle successive analisi, evitando ripetizioni.

3.3 Ricerca degli Utenti Correlati

Il codice sviluppato per l'estrazione di informazioni sugli utenti di X è progettato per identificare e analizzare gli utenti correlati a quelli già individuati in precedenti ricerche. L'obiettivo del processo è l'estrazione automatizzata di informazioni utili per ogni utente target, salvandole in un database per ulteriori analisi.
Inizialmente, il codice si connette a un database MongoDB e carica un elenco di utenti già profilati dalla collezione `users_info`. Utilizzando Selenium, il sistema automatizza l'accesso alla piattaforma di X, e per ogni utente target individua gli utenti correlati (tramite una funzione dedicata) navigando nei rispettivi profili e applicando verifiche sull'esistenza e limitazioni di accesso.
Ogni utente correlato individuato viene quindi sottoposto a un'analisi dettagliata attraverso il modulo BeautifulSoup, che filtra e struttura i dati del profilo (come numero di follower, bio, verificato) per il salvataggio finale nella collezione del database. Il sistema inoltre cattura contenuti multimediali come immagini e video quando presenti nei tweet degli utenti. In conclusione, il codice gestisce in modo modulare sia l'estrazione che la strutturazione di queste informazioni per garantirne l'inserimento efficiente nel database, assicurando l'aggiornamento e la validità dei dati raccolti.

4 MongoDB

MongoDB è un database NoSQL orientato ai documenti, progettato per gestire grandi quantità di dati non strutturati o semi-strutturati. A differenza dei database relazionali tradizionali (RDBMS), MongoDB non utilizza tabelle e schemi rigidi, ma archivia i dati in documenti JSON-like chiamati *BSON* (Binary JSON), che permettono una maggiore flessibilità nella gestione delle informazioni.
MongoDB è organizzato in:

- **Database**: è l'unità più grande che contiene una collezione di dati correlati.
- **Collezioni**: ogni database può contenere una o più collezioni, l'equivalente delle tabelle in un database relazionale. Tuttavia, una collezione non ha uno schema fisso.
- **Documenti**: all'interno di una collezione, i dati vengono memorizzati sotto forma di documenti BSON. Ogni documento è una struttura JSON-like che contiene campi chiave-valore. Questa flessibilità permette che i documenti all'interno della stessa collezione possano avere campi differenti.

MongoDB supporta le operazioni CRUD (Create, Read, Update, Delete) e permette la memorizzazione di dati complessi, come array o documenti nidificati. È particolarmente adatto per applicazioni che devono gestire dati eterogenei o in rapido cambiamento, come il web scraping o i big data.
Nel codice Python, MongoDB viene utilizzato tramite la libreria `pymongo`, che consente di interagire facilmente con il database.

5 Risultati

Una volta eseguita la ricerca, effettuato lo scraping e il parsing dei dati estratti questi ultimi vengono salvati su MongoDB. Prima di essere salvati, i risultati vengono organizzati, durante il processo di parsing, in oggetti JSON che poi potranno essere archiviati direttamente.

5.1 Tweet

Nel Database ci sono varie collection che contengono documenti relativi ai tweet estratti. Ogni collection è relativa ad un diverso gruppo hacker a cui tali tweet fanno riferimento.
Il formato degli oggetti salvati è il seguente:

```
{
  "username": "string",
  "username_tag": "string",
  "data_pubblicazione": "string",
  "contenuto": "string",
  "commenti": "string",
  "repost": "string",
  "like": "string",
  "visualizzazioni": "string",
  "url": "string"
}
```

Esempio:

```
{
  "username": "lazarus",
  "username_tag": "@laz_tboi",
  "data_pubblicazione": "7 giu 2024",
  "contenuto": "im gonna beat up isaac on downpour 1 everyone be there",
  "commenti": "1",
  "repost": "1",
  "like": "3",
```

```
  "visualizzazioni": "54",
  "url": "https://x.com/laz_tboi/status/1798945030469787863"
}
```

5.2 Informazioni Utente

La collection 'Users_info' nel database contiene documenti relativi alle informazioni principali estratte dal profilo di ogni utente analizzato.
Il formato degli oggetti salvati è il seguente:

```
{
  "username_tag": "@string",
  "verified": "bool",
  "num_post": "string",
  "following": "string",
  "followers": "string",
  "job": "string",
  "location": "string",
  "subscription": "string",
  "birth": "string",
  "website": "string"
}
```

Esempio:

```
{
  "username_tag": "@CNN",
  "verified": true,
  "num_post": "422.743",
  "following": "1.078",
  "followers": "62,7 Mln",
  "job": null,
  "location": null,
  "subscription": "febbraio 2007",
  "birth": null,
  "website": "cnn.com"
}
```

6 Repository Github

Di seguito il link alla Repository Github.