Rilevatore di Verbi HTTP

Per realizzare un rilevatore di verbi HTTP è necessario importare la libreria “http.client” che fa parte della libreria standard di Python e consente di gestire le connessioni HTTP a basso livello. Attraverso questa libreria è possibile creare connessioni verso server web, inviare richieste con metodi specifici e ricevere le relative risposte, analizzando i codici di stato e gli header.

Successivamente, il programma chiede all’utente di inserire l’indirizzo IP del server target e il numero di porta da utilizzare per la connessione. Se l’utente non specifica alcun valore per la porta, viene automaticamente utilizzata la porta 80, che è quella convenzionalmente impiegata per le connessioni HTTP non cifrate. Questo permette di eseguire il test anche su server standard senza dover indicare ogni volta il numero di porta.

```
1 import http.client
2
3 host = input("IP target: ")
4 port = input("Port target (default. 80): ")
5
```

Il programma definisce poi un insieme di metodi HTTP da testare sul server e crea due variabili: una per conservare l’elenco dei metodi accettati e l’altra per contarli. Il blocco centrale del codice è racchiuso in una struttura try, necessaria per gestire in modo pulito eventuali errori legati alla connessione, ad esempio quando il server rifiuta la comunicazione.

Per ogni metodo HTTP della lista, il codice apre una connessione al server specificato, invia una richiesta utilizzando quel metodo verso il percorso radice e attende la risposta. Dopo aver ricevuto la risposta, il codice stampa a schermo lo status code, che rappresenta il risultato dell’elaborazione della richiesta da parte del server. Se lo status code è 200, il che indica una risposta positiva con successo, il metodo viene considerato accettato e viene aggiunto all’elenco dei metodi supportati, incrementando anche il contatore.

```
6 metods= ["GET", "POST", "PUT", "OPTIONS", "HEAD"]
7 accepteds = []
8 counter = 0
9
10 if(port == ""):
11         port =80
12
13 try:
14         connection = http.client.HTTPConnection(host, port)
15         for metod in metods:
16                 connection.request(metod, '/')
17                 response = connection.getresponse()
18                 print(metod, "response: ", response.status)
19                 connection.close()
20                 if(response.status==200):
21                         accepteds.append(metod)
22                         counter+=1
```

Una volta completato il ciclo di test, il programma verifica se almeno un metodo è stato accettato. Se nessuno ha ricevuto risposta positiva, viene mostrato un messaggio che indica che nessun metodo è stato accettato dal server. Al contrario, se uno o più metodi sono stati accettati, questi vengono stampati in una riga unica, separati da virgole, per facilitare la lettura del risultato. Il codice termina con una gestione dell'eccezione ConnectionRefusedError, che intercetta eventuali errori nel tentativo di stabilire la connessione e mostra un messaggio di errore in caso di fallimento, evitando che il programma si interrompa bruscamente.

```
23          if(counter == 0):
24                  print("\nAccepted metods for", host,":", "NOONE")
25          else:
26                  print("\nAccepted metods for", host,":", ", ".join(accepteds))
27  except ConnectionRefusedError:
28          print("Failed Connection")
```

Per poter testare il codice, ho avviato un server HTTP con python usando il codice:
python3 -m http.server 8080

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

e subito dopo ho avviato il programma inserendo come host l'IP di localhost "127.0.0.1" e come porta ho messo la porta che ho dichiarato quando ho avviato il server con python:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python httpMetods.py
IP target: 127.0.0.1
Port target (default. 80): 8080
GET response:  200
POST response:  501
PUT response:  501
OPTIONS response:  501
HEAD response:  200

Accepted metods for 127.0.0.1 : GET, HEAD
```

Ottenendo lo stesso riscontro sul server:

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
127.0.0.1 - - [13/May/2025 12:05:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [13/May/2025 12:05:59] code 501, message Unsupported method ('POST')
127.0.0.1 - - [13/May/2025 12:05:59] "POST / HTTP/1.1" 501 -
127.0.0.1 - - [13/May/2025 12:05:59] code 501, message Unsupported method ('PUT')
127.0.0.1 - - [13/May/2025 12:05:59] "PUT / HTTP/1.1" 501 -
127.0.0.1 - - [13/May/2025 12:05:59] code 501, message Unsupported method ('OPTIONS')
127.0.0.1 - - [13/May/2025 12:05:59] "OPTIONS / HTTP/1.1" 501 -
127.0.0.1 - - [13/May/2025 12:05:59] "HEAD / HTTP/1.1" 200 -
```

Ho ottenuto questo risultato perché di default, il server "http.server" di Python accetta solo GET e HEAD