Web Security: curl e requests

Leonardo Taccari
<s1069964@studenti.univpm.it>

Sommario

Un po' di pratica!

curl: richiesta HTTP GET

- curl è un client HTTP
- Vediamo una semplice richiesta HTTP GET!
- Per le prove seguenti utilizzeremo httpbin.org che è un server HTTP che supporta molti metodi HTTP e ci stampa la richiesta che abbiamo fatto formattata in JSON.

```
$ curl http://httpbin.org/get
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Host": "httpbin.org",
    "User-Agent": "curl/8.12.0",
    "X-Amzn-Trace-Id": "Root=1-67ac5b9f-702782b106ebb1af7a6bfcf5"
  },
  "origin": "5.77.65.203",
  "url": "http://httpbin.org/get"
}
```

Un po' di pratica!

JSON (JavaScript Object Notation)

- JSON (JavaScript Object Notation) è un formato utilizzato per lo scambio di dati
- Deriva dal linguaggio di programmazione JavaScript
- È utilizzato in moltissimi servizi Web per scambiare dati
- Il suo tipo MIME [1] è `application/json`

[1] Che ritroveremo negli header HTTP `Accept:` per le richieste HTTP e `Content-Type:` per le risposte HTTP.

Un po' di pratica! I

requests: richiesta HTTP GET

- Requests è una libreria HTTP Python
- Vediamo la stessa richiesta utilizzando requests!

```
$ python3
Python 3.12.9 (main, Feb 10 2025, 00:16:27) [GCC 12.4.0] on netbsd10
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> response = requests.get("http://httpbin.org/get")
>>> # response conterrà la risposta HTTP ed altre meta-informazioni
>>> # response.text è il body della risposta HTTP decodificato in str
>>> print(response.text)
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br, zstd",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.32.3",
    "X-Amzn-Trace-Id": "Root=1-67ac5f26-5526d7f1254e39cc6c30b253"
  },
  "origin": "5.77.65.203",
  "url": "http://httpbin.org/get"
}
```

- Curiosiamo altri campi della `response`!

Un po' di pratica! II

requests: richiesta HTTP GET

```
>>> response.status_code  # contiene lo status code della risposta HTTP
200
>>> response.reason  # contiene lo status text della risposta HTTP
'OK'
>>> response.headers  # dict che contiene tutti gli HTTP header della response
{'Date': 'Wed, 12 Feb 2025 08:48:13 GMT', 'Content-Type': 'application/json', ... }
```

- Diamo un'occhiata alla documentazione ufficiale ed al quickstart in `https://requests.readthedocs.io/`

Un po' di pratica! I

curl: richiesta HTTP GET con parametri

- Possiamo passare parametri alle richieste HTTP in modo da ottenere una particolare versione della risorsa
- I parametri vengono passati tramite la query
- La query è preceduta da un ? e contiene una coppia di chiave-valori

```
$ curl http://httpbin.org/get?chiave=valore
{
  "args": {
    "chiave": "valore"
  },
  "headers": {
    ...
  },
  ...
  "url": "http://httpbin.org/get?chiave=valore"
}
```

- Se vogliamo passare più chiavi-valori possiamo separiamo le chiavi con il carattere &, ad esempio
  `curl 'http://httpbin.org/get?chiave1=foo&chiave2=bar` [2]

Un po' di pratica! II

curl: richiesta HTTP GET con parametri

- In alternativa possiamo utilizzare le opzioni `--get` e `--data` di `curl`, in modo da non dover preoccuparci di preparare la query string noi nell'URL.
- Dobbiamo specificare `--get` per specificare che vogliamo una richiesta HTTP con il verbo GET perché quando passiamo `--data` curl assume che vogliamo usare il verbo POST altrimenti.

```
$ curl --get --data-urlencode 'chiave=valore con spazi e == e ?' http://httpbin.org/get
{
  "args": {
    "chiave": "valore con spazi e == e ?"
  },
  "headers": {
    ...
  },
  ...,
  "url": "http://httpbin.org/get?chiave=valore+con+spazi+e+%3d%3d+e+%3f"
}
```

- Come facciamo se vogliamo inserire un = o ? o altri caratteri speciali nella query?

Un po' di pratica! III

curl: richiesta HTTP GET con parametri

- Utilizziamo l'URL encoding!
- In curl, invece di usare `--data` basta che utilizziamo `--data-urlencode`:

[2] Abbiamo messo l'URL tra apici singoli ' così che caratteri come & non vengono interpretati dalla shell... nel dubbio mettete sempre ciò tra apici singoli!

Un po' di pratica! I

requests: richiesta HTTP GET con parametri

- In requests possiamo usare l'argomento opzionale `params` per passare un dizionario con le chiavi-valori che vogliamo:

```
>>> import requests
>>> response = requests.get("http://httpbin.org/get", params={"chiave": "valore"})
>>> print(response.text)
{
  "args": {
    "chiave": "valore"
  },
  ...
  "url": "http://httpbin.org/get?chiave=valore+con+%3D%3D+e+%3F"
}
```

- In requests l'URL encoding avviene automaticamente

Un po' di pratica! I

curl: richiesta HTTP con header HTTP personalizzato

- Possiamo passare degli header HTTP personalizzati per avere una rappresentazione diversa della risorsa o autenticarci con delle credenziali al server HTTP
- Possiamo fare ciò con l'opzione `--header`

```
$ curl --header 'X-Custom: value' http://httpbin.org/get
{
  "args": {},
  "headers": {
    ...,
    "X-Custom": "value"
  },
  ...
  "url": "http://httpbin.org/get"
}
```

Un po' di pratica! I

requests: richiesta HTTP con header HTTP personalizzato

- In requests, per passare degli header HTTP personalizzati possiamo usare l'argomento opzionale `headers` che accetta un dizionario con HTTP header e corrispondenti valori:

```
>>> import requests
>>> response = requests.get("http://httpbin.org/get", headers={"X-Custom": "value"})
>>> print(response.text)
{
  "args": {},
  "headers": {
    ...,
    "X-Custom": "value"
  },
  ...,
  "url": "http://httpbin.org/get"
}
```

Conclusioni

- Abbiamo visto come fare richieste HTTP con `curl` e `requests`
- Abbiamo anche dato un'occhiata come avviene lo scambio di informazioni utilizzando JSON
- Mettendo insieme ciò possiamo ora "scriptare" le nostre richieste HTTP!

Riferimenti

- Materiale Didattico del Portale di allenamento delle Olimpiadi Italiane di Cybersicurezza
- MDN Web Docs
- curl
- Requests: HTTP for Humans