

CMPE-279
Assignment-3

Team Members:

Shanmukha Yaswanth Reddy Kallam	015998840
Purna Sai Mahesh Goud Palagani	015999308

1.

Describe the SQLi attack you used, how did you cause the user table to be dumped? What was the input string you used?



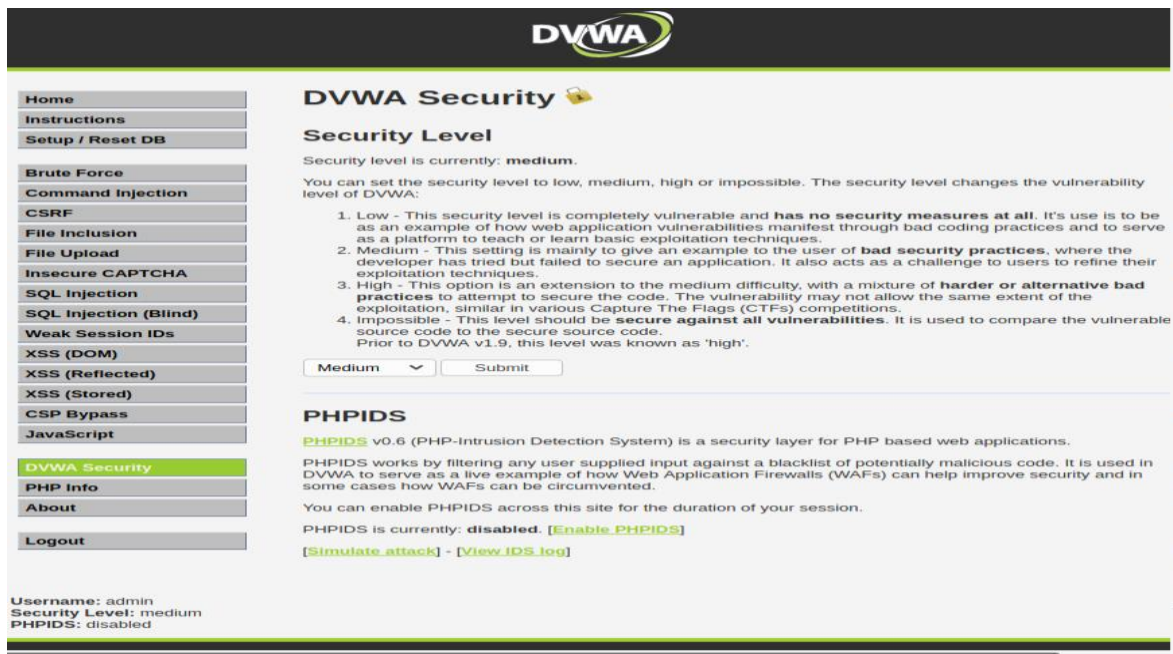
Firstly, We setup the security level of DVWA (Damn Vulnerable Web App) to “low” .



Now we pass 1' or 1=1# to satisfy the queries and display the list of users in the database. It then displayed the “FIRST NAME” and “SURNAME” of all the users in the database.

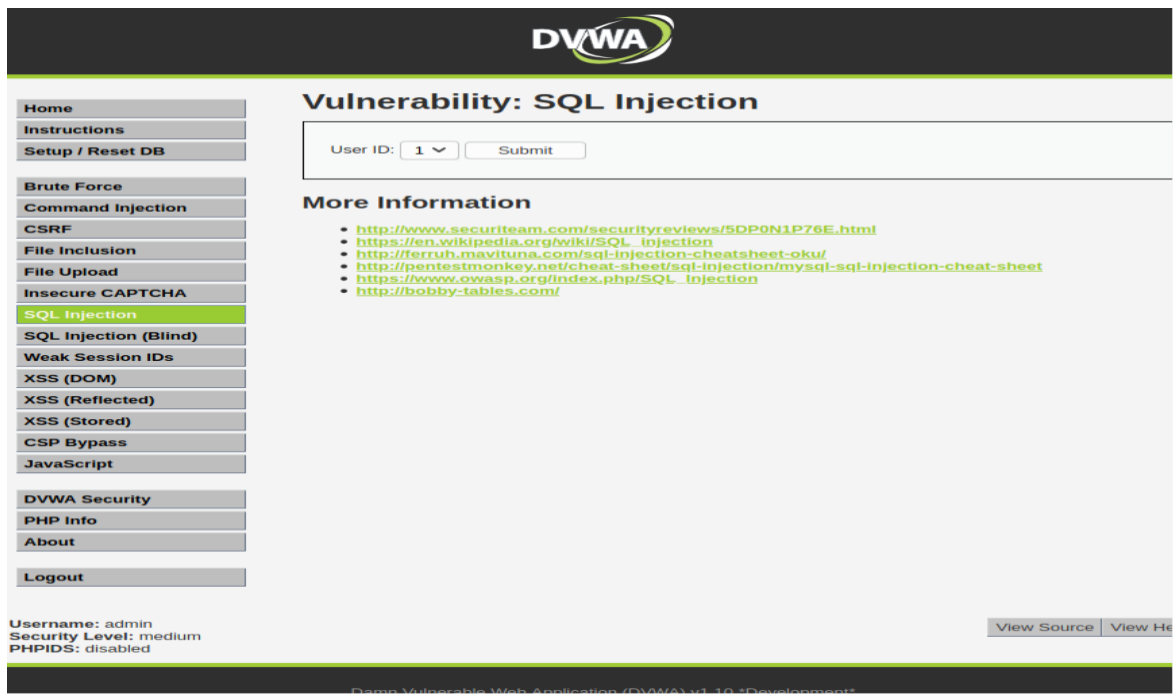
2.

If you switch the security level in DVWA to “medium”, does the SQLi attack still works?



The screenshot shows the DVWA Security page. The left sidebar contains a menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security (highlighted), PHP Info, About, and Logout. The main content area is titled "DVWA Security" and "Security Level". It states the current security level is "medium". Below this, it explains the security levels: 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'. Below the explanation, there is a dropdown menu set to "Medium" and a "Submit" button. The "PHPIDS" section is also visible, stating it is currently disabled and providing links to enable it, simulate an attack, and view the log. At the bottom, the status bar shows: Username: admin, Security Level: medium, PHPIDS: disabled.

Now, if we check the same query after changing the security level of DVWA to “medium” then the SQL injection attack didn’t work.

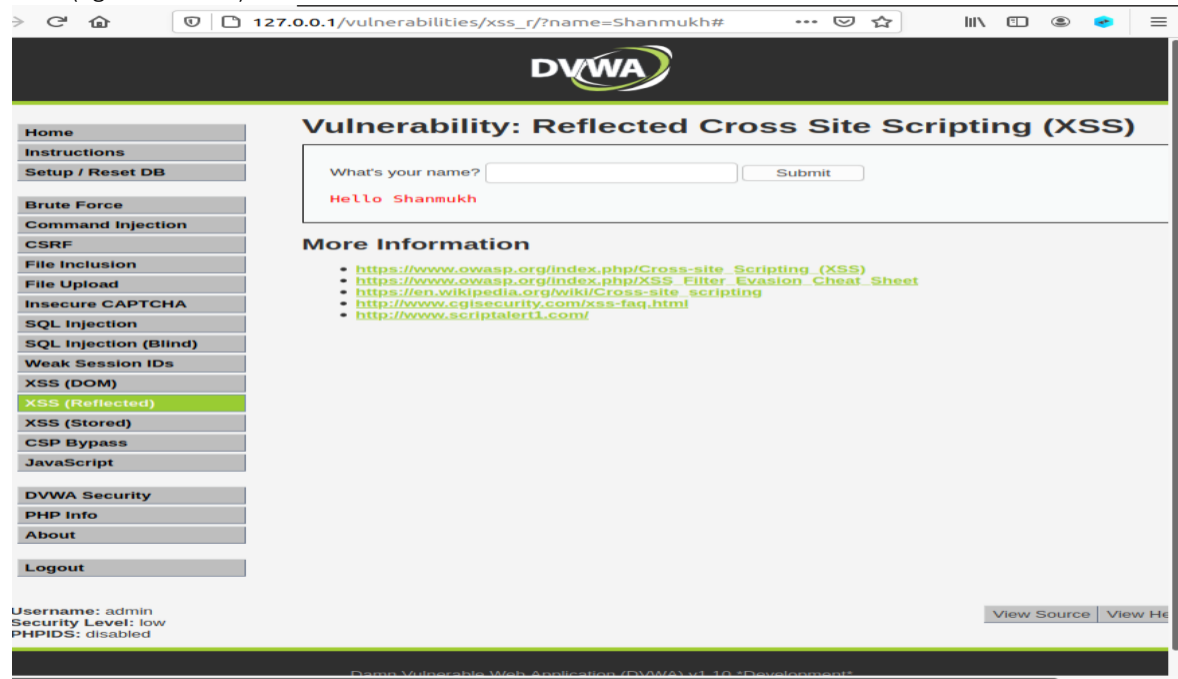


The screenshot shows the DVWA Vulnerability: SQL Injection page. The left sidebar is the same as the previous screenshot, with "SQL Injection" highlighted. The main content area is titled "Vulnerability: SQL Injection". It contains a form with a "User ID:" label, a dropdown menu set to "1", and a "Submit" button. Below the form, there is a "More Information" section with a list of links to external resources: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, https://en.wikipedia.org/wiki/SQL_injection, <http://feruh.mavikuna.com/sql-injection-cheatsheet-oku/>, <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>, https://www.owasp.org/index.php/SQL_injection, and <http://bobby-tables.com/>. At the bottom right, there are links for "View Source" and "View Help". The status bar at the bottom shows: Username: admin, Security Level: medium, PHPIDS: disabled.

3.

Describe the reflected XSS attack you used, how did it work?

Now, set the security level to low and select XSS(Reflected) from the available options and enter input name here (eg: Shanmukh) .



A message saying “Hello Shanmukh” pops on the screen.

4.



If we change the security level to medium and give input `<script>("Hello From Input")</script>` . XSS attack doesn't work the input was processed as characters but not as code/program.

