COMP 307 – Lab Assignment #6

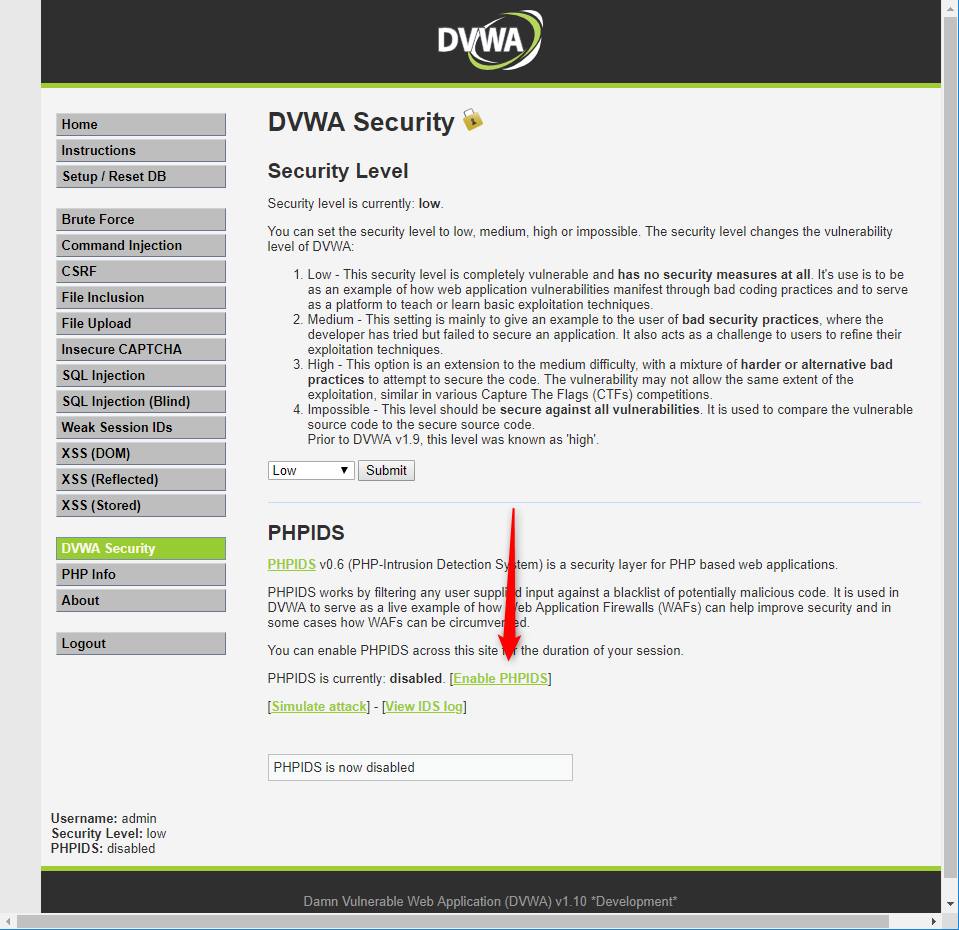
Kevin Ma – 300867968

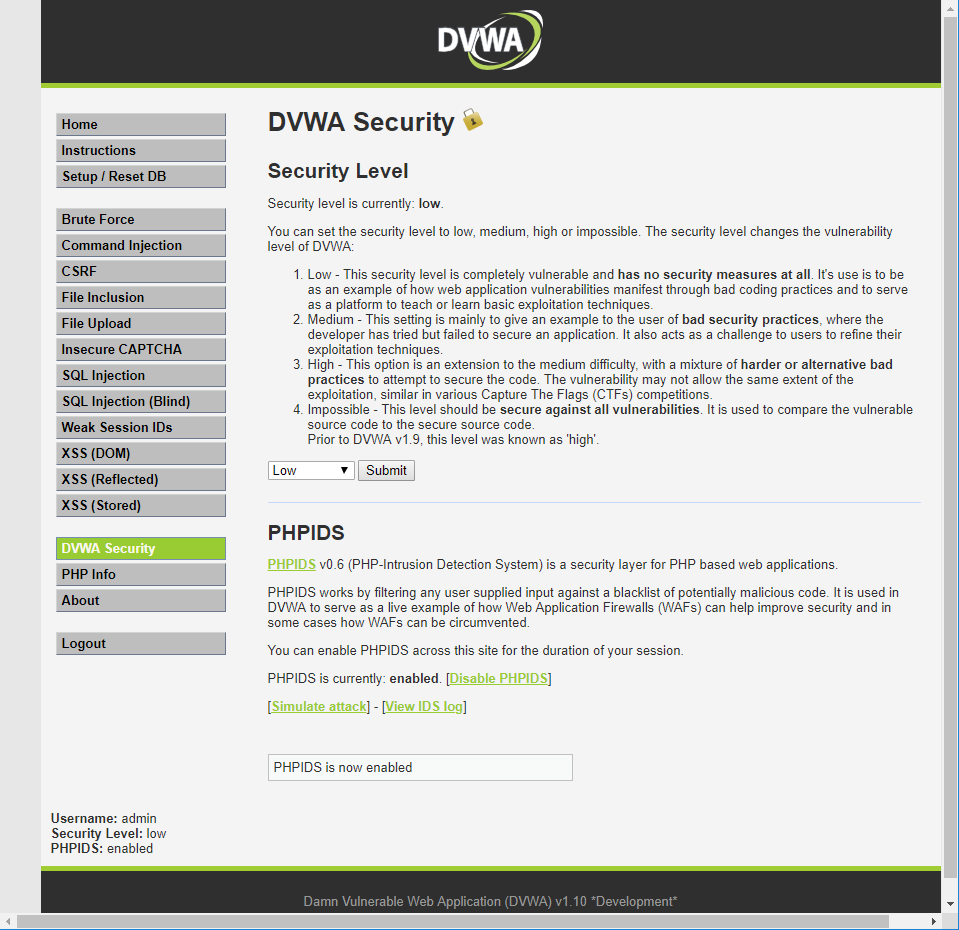
Ostap Hamanryk – 300836326

October 15, 2018

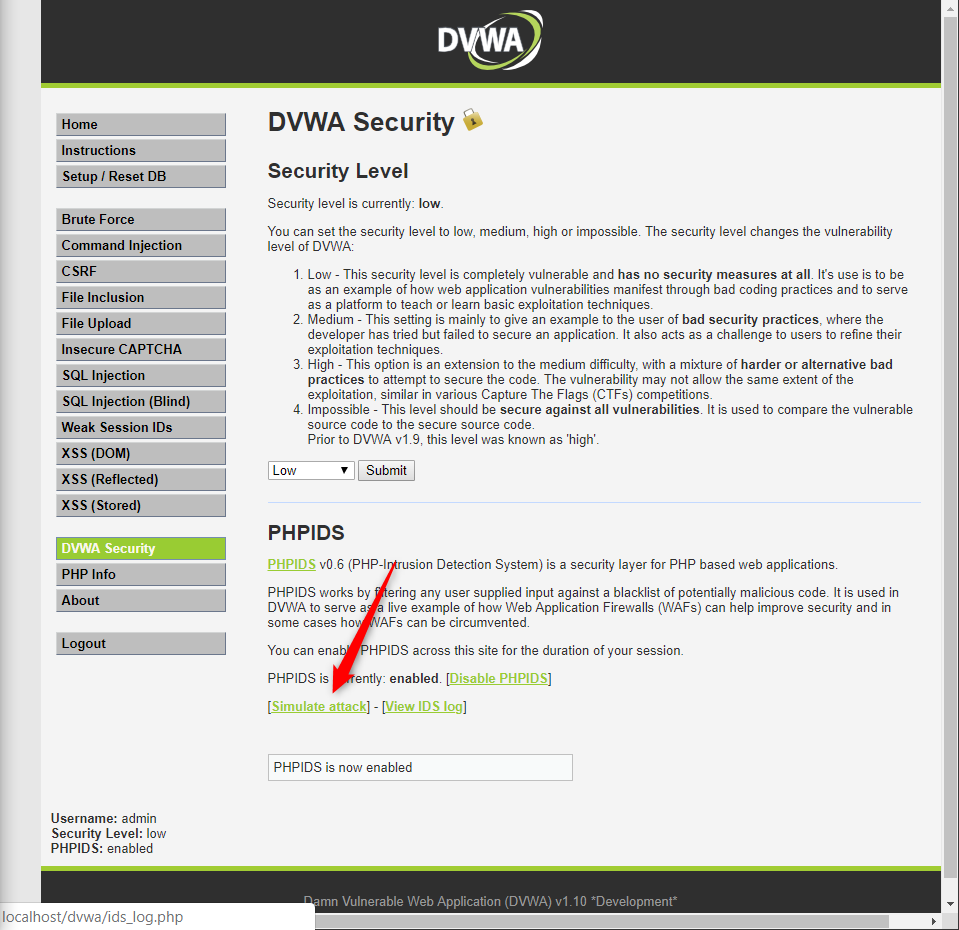
# A. Detect a hacking attack using a PHPIDS

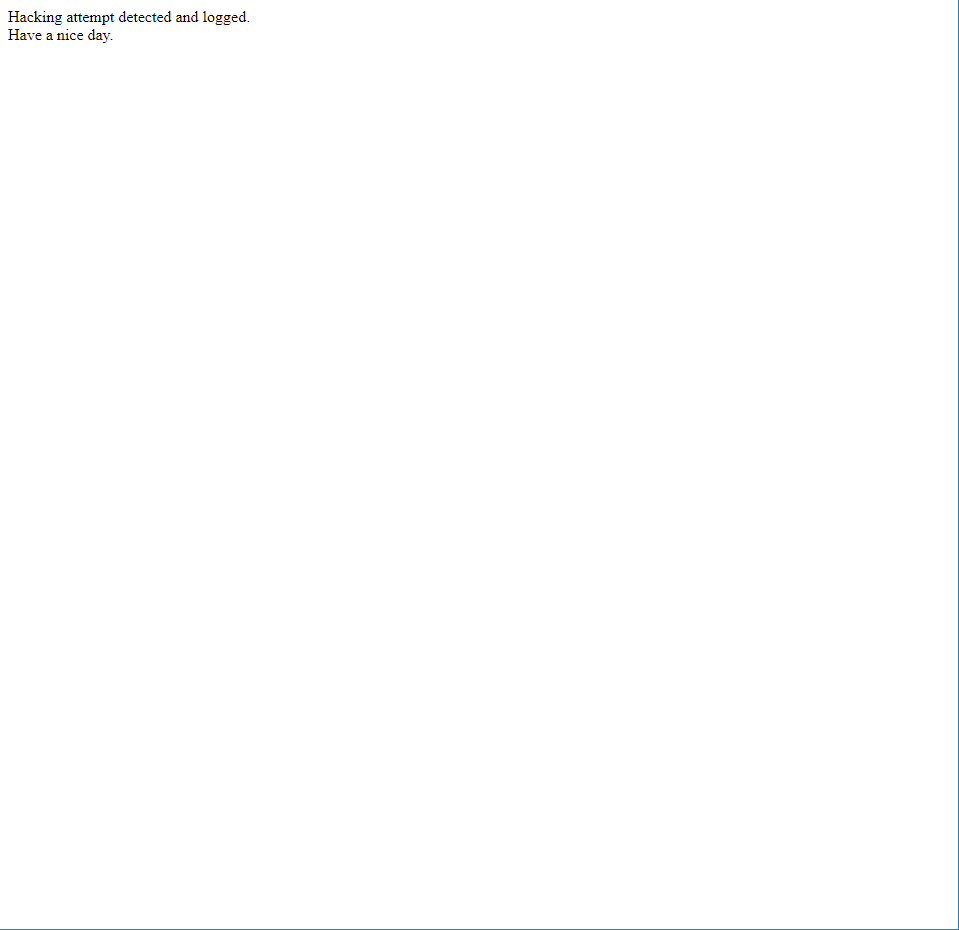
* 1. Enable PHPIDS



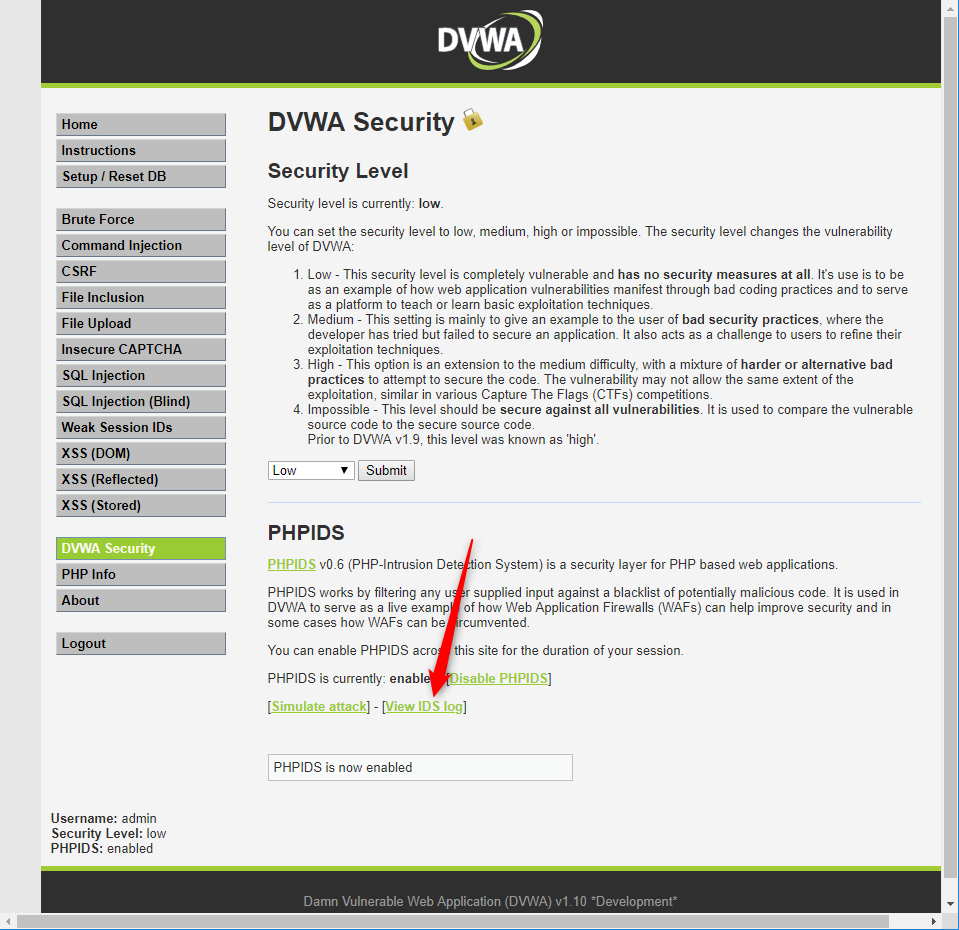


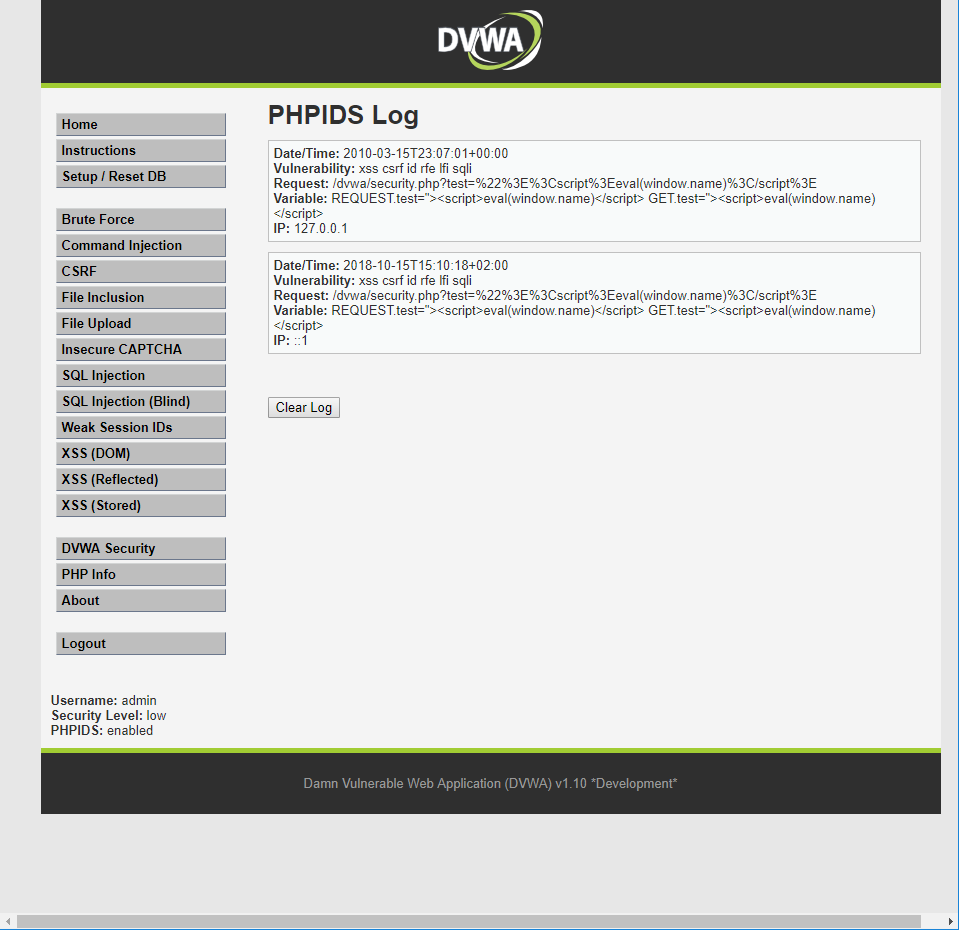
* 1. Simulate attack



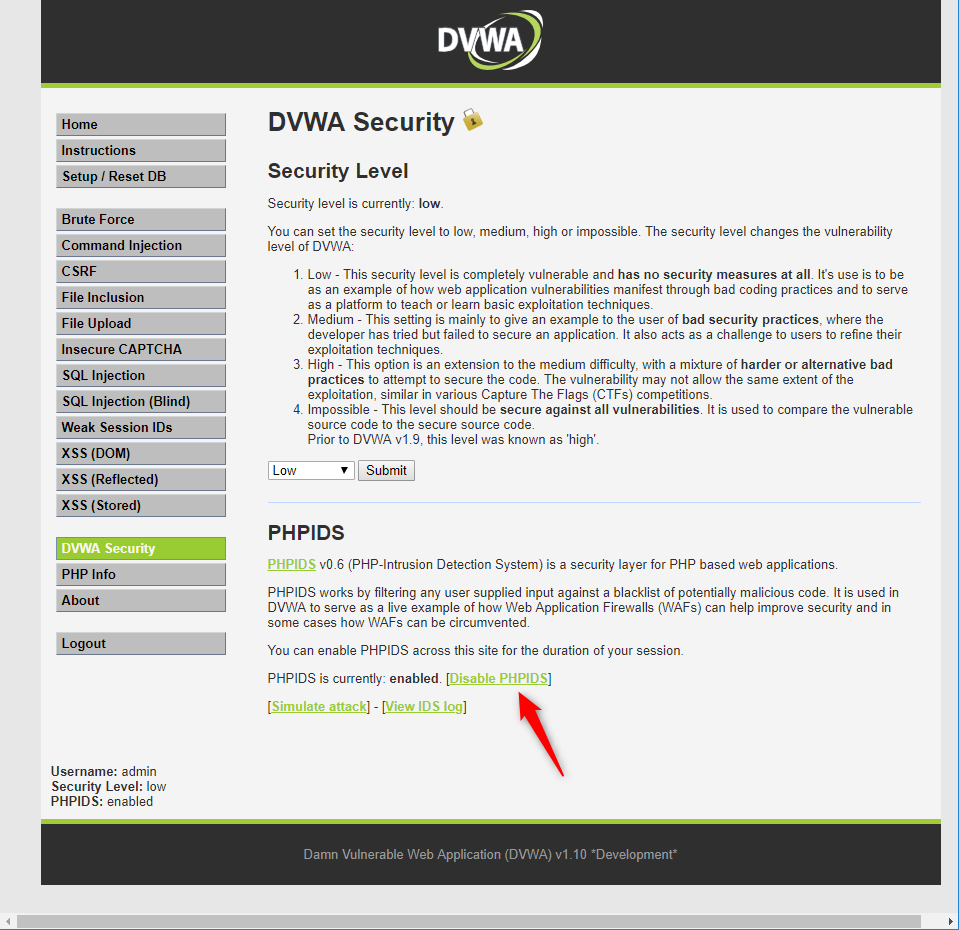


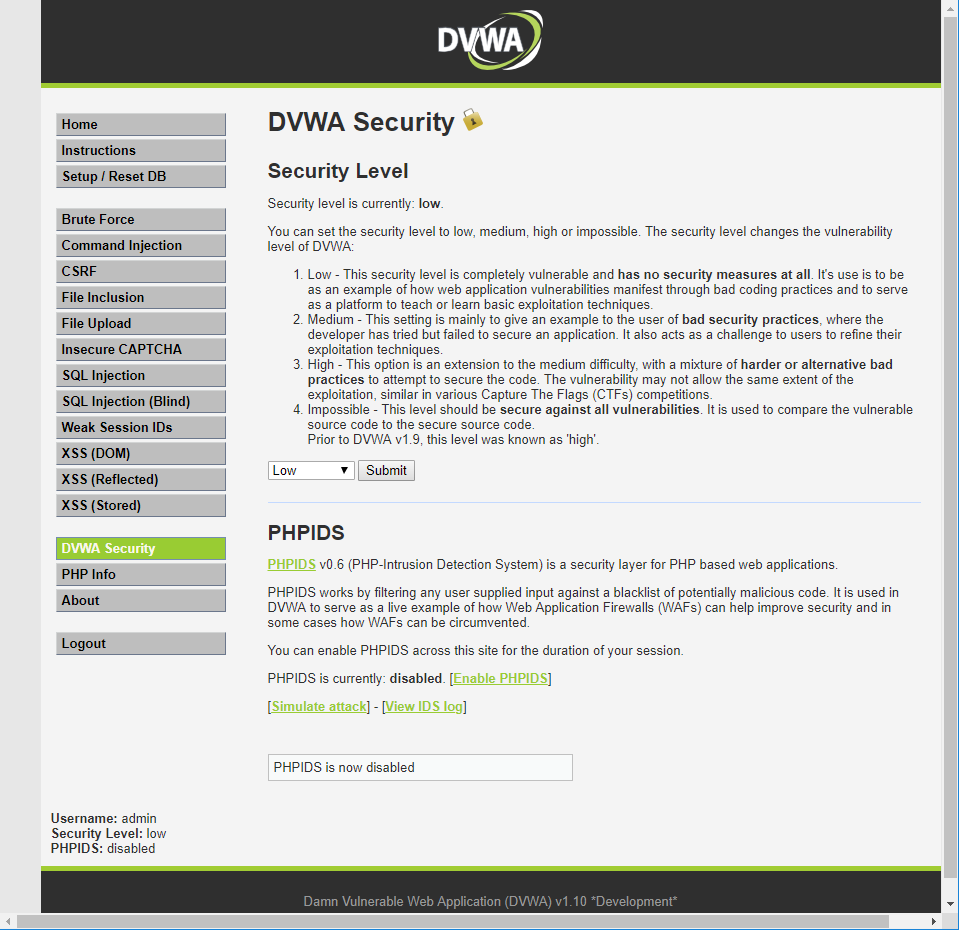
* 1. See logs



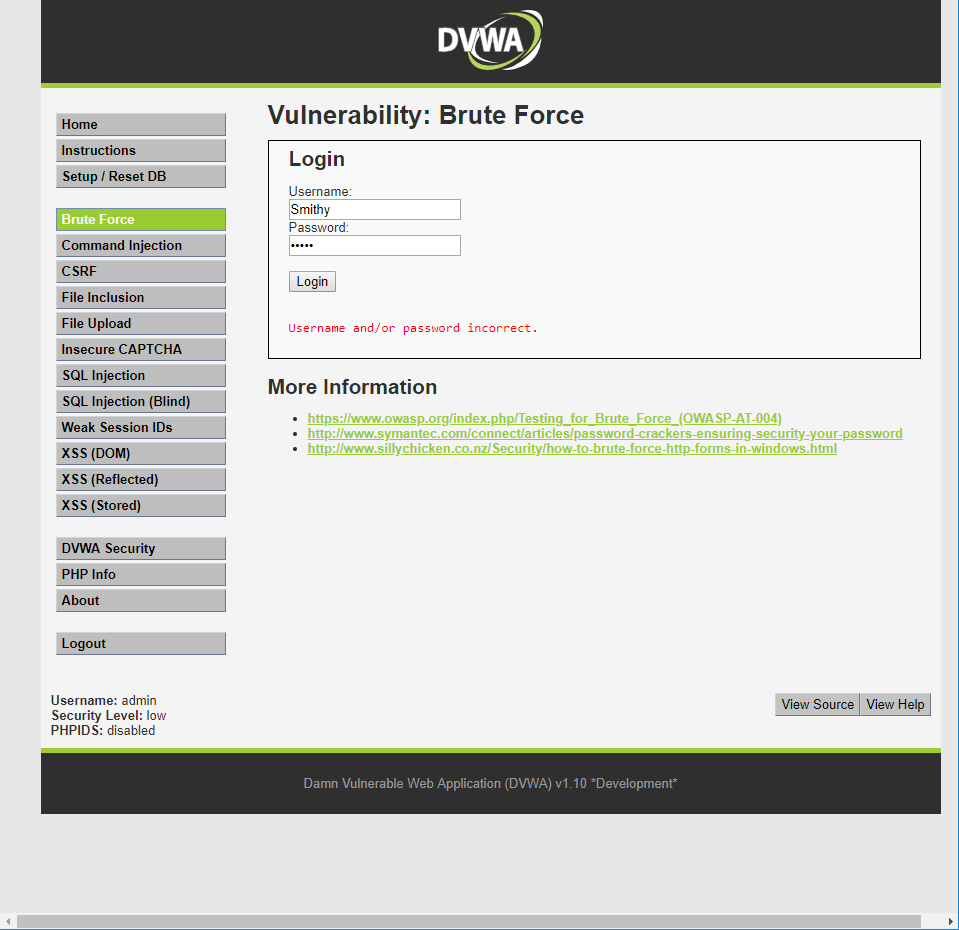


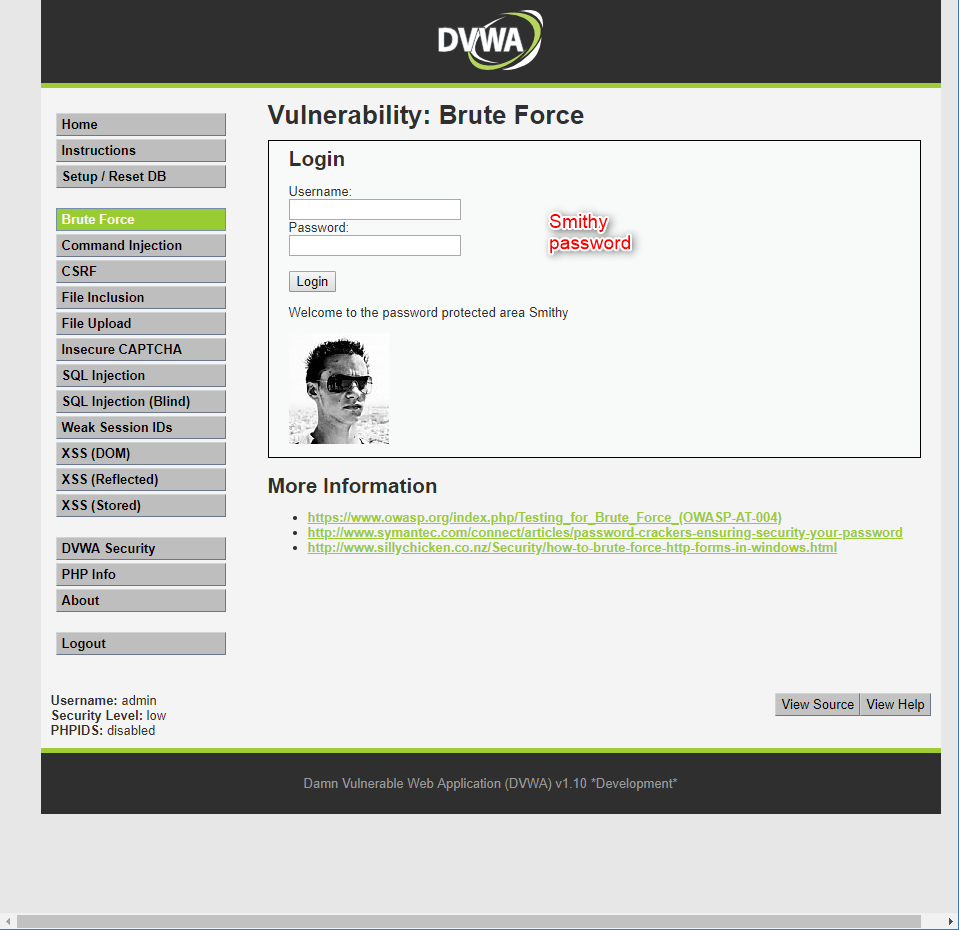
* 1. Disable PHPIDS





# B. Brute Force



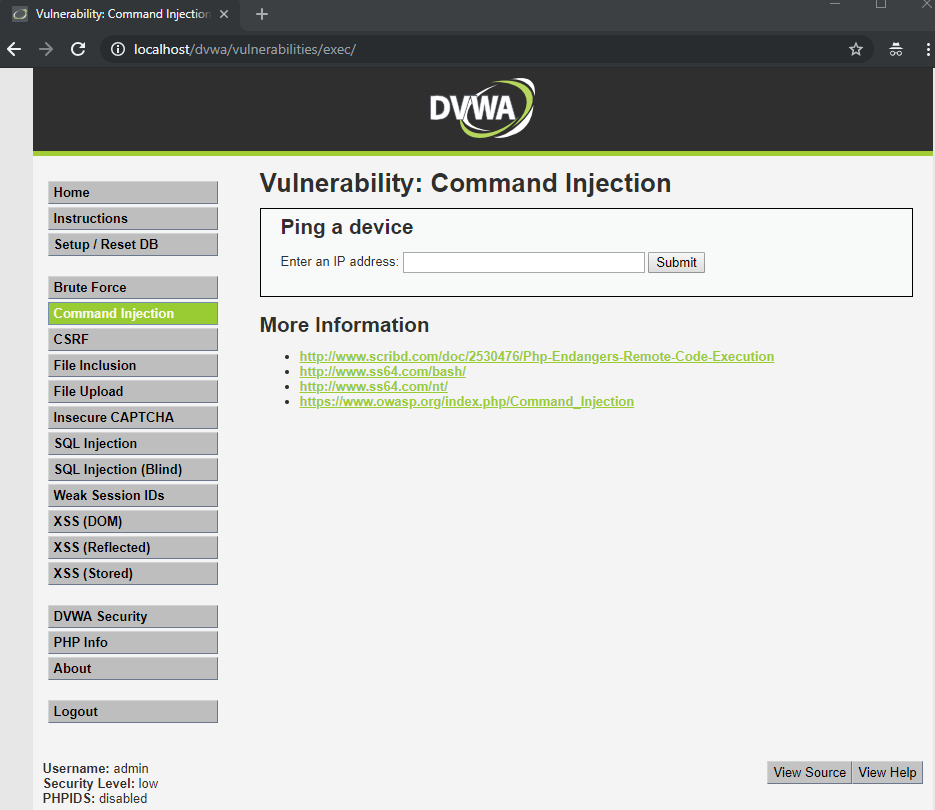


There are differences for brute forcing depending on the security level set for the DVWA application:

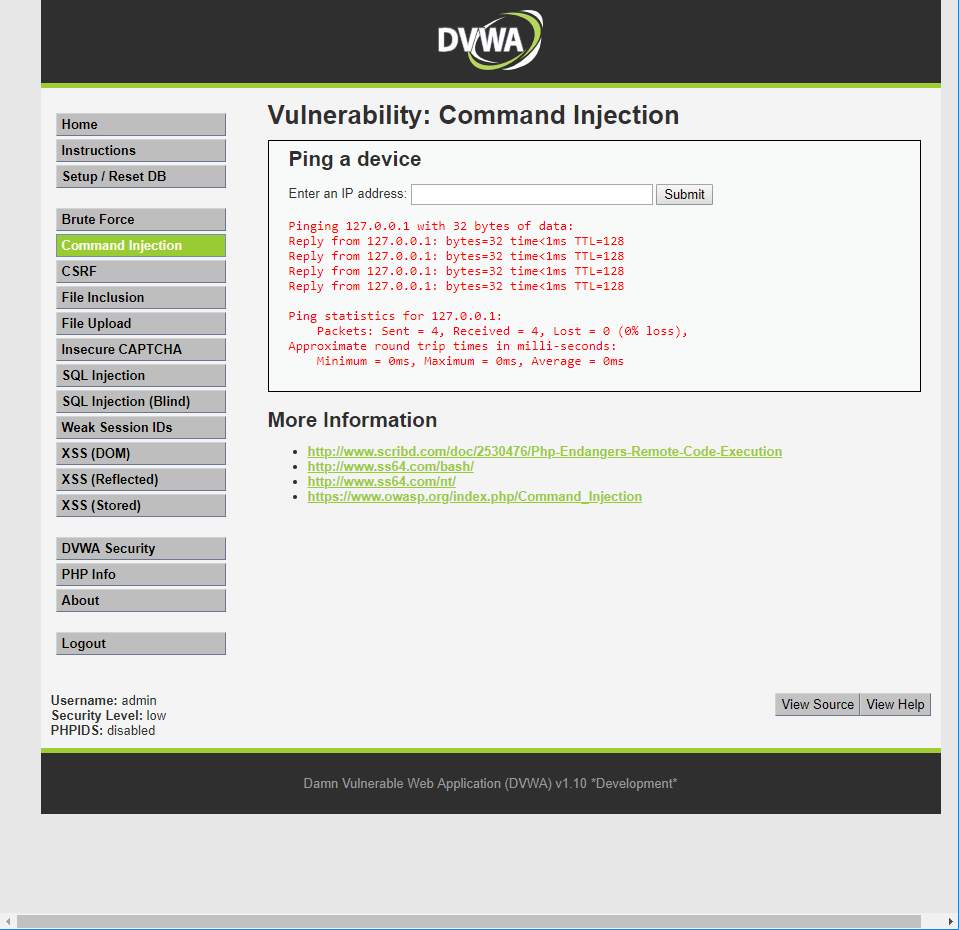
* Low
  + - Simply returns error message when invalid login provided
      * Username and/or password incorrect.
* Medium
  + - Sanitizes the username and password fields
      * Three of the five most common website attacks—SQL injection, cross-site scripting (XSS), and remote file inclusion (RFI)—all are exploited by data sent to Web server by the end user
        + When the end user is a good guy, the data he sends the server is relevant to his interaction with the website. But when the end user is a hacker, she can exploit this mechanism to send the Web server input which is deliberately constructed to escape the legitimate context and execute unauthorized actions.
      * Input sanitization describes cleansing and scrubbing user input to prevent it from jumping the fence and exploiting security holes. But thorough input sanitization is hard. While some vulnerable sites simply don’t sanitize at all, others do so incompletely, lending their owners a false sense of security.
    - Introduces a delay of 2 seconds between retries to prevent hacker from spamming mass number of requests to brute force guess the password
      * sleep( 2 );
* High
  + - Check Anti-CSRF token
      * Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request.
      * In short, CSRF abuses the trust relationship between browser and server. This means that anything that a server uses to establish trust with a browser (e.g., cookies, but also HTTP/Windows Authentication) is exactly what allows CSRF to take place - but this only the first piece for a successful CSRF attack.
      * The second piece is a web form or request which contains parameters predictable enough that an attacker could craft his own malicious form/request which, in turn, would be successfully accepted by the target service. Then, usually through social engineering or XSS, the victim would trigger that malicious form/request submission while authenticated to the legitimate service. This is where the browser/server trust is exploited.
    - Sanitizes the username and password input fields
    - Introduces a random delay between 0 and 3 seconds between retries to prevent hacker from spamming mass number of requests to brute force guess the password
      * This would prevent the hacker from figuring out the delay and being able to code their hacking software to retry every 2 seconds
    - Generate Anti-CSRF token
* Impossible
  + - Check Anti-CSRF token
    - Uses POST requests instead of GET requests
      * Using GET requests would pass the user’s credentials as parameters in the URL
      * Using POST requests passes the user’s credentials inside of the request body which is not visible to the naked eye
    - Added default database lockout parameters
      * if user has 3 total failed logins, he/she will be locked out of his/her account for 15 minutes before he/she can attempt to login again
    - When user fails to login, notify wrong credentials or account has been locked out.
    - Generate Anti-CSRF token

# C. Command Injection

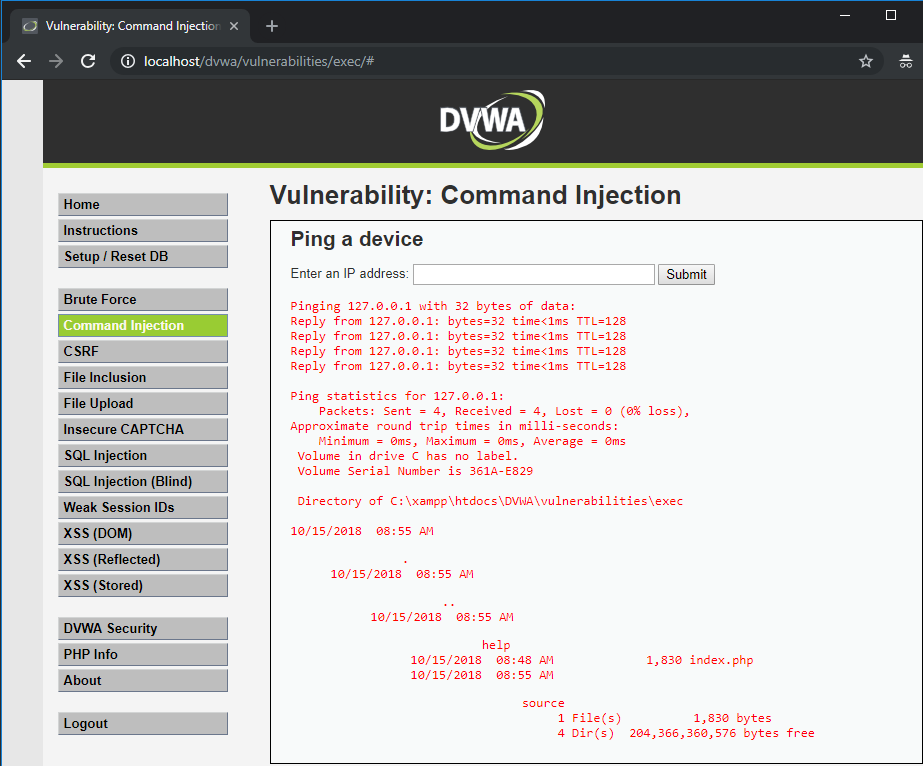
1. Set Security Low
2. Go to Command Injection



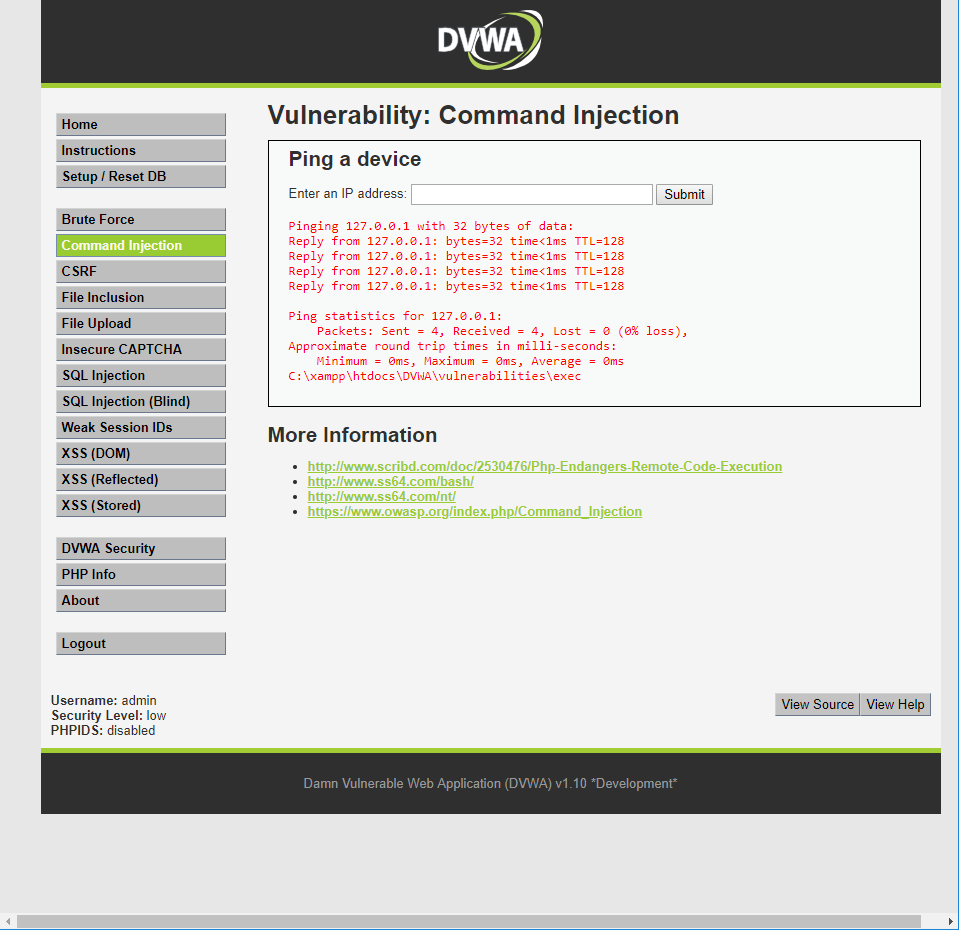
1. Enter '127.0.0.1' and take snapshot



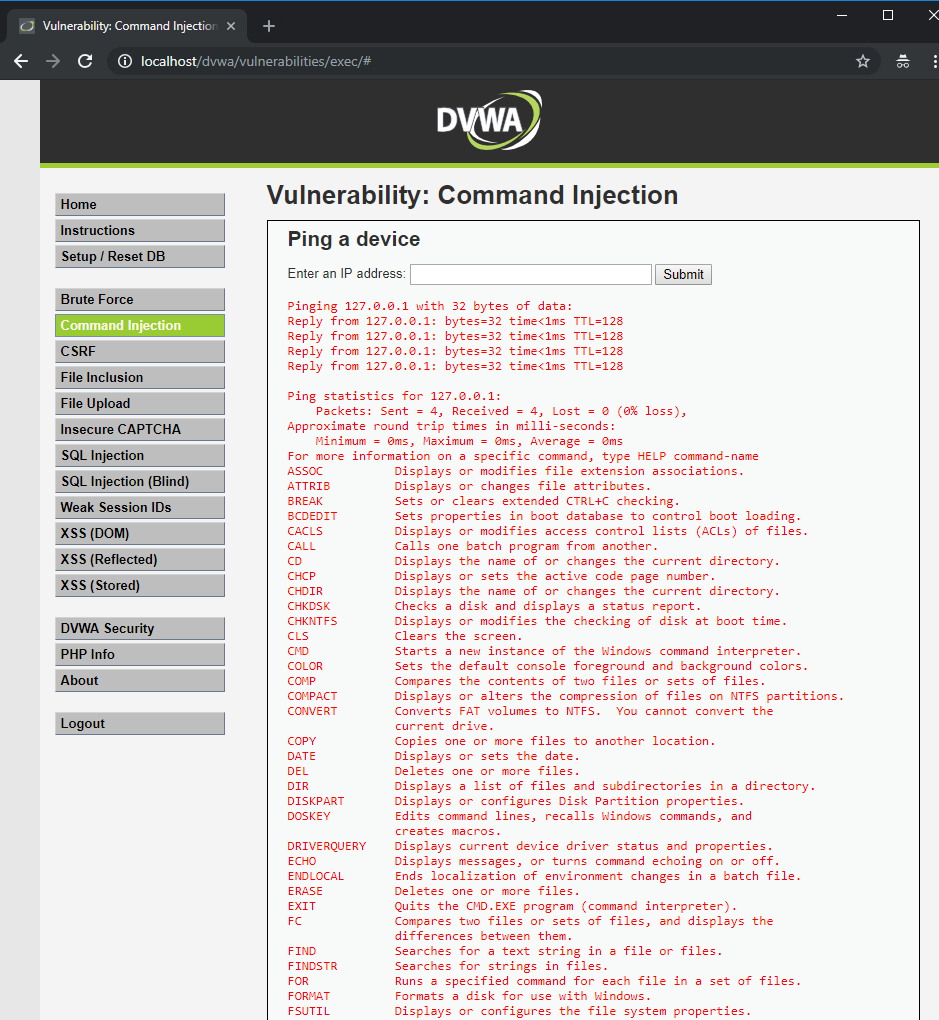
1. Enter '127.0.0.1 && dir' and take snapshot



1. Enter '127.0.0.1 && cd' and take snapshot



1. Enter '127.0.0.1 && help' and take snapshot

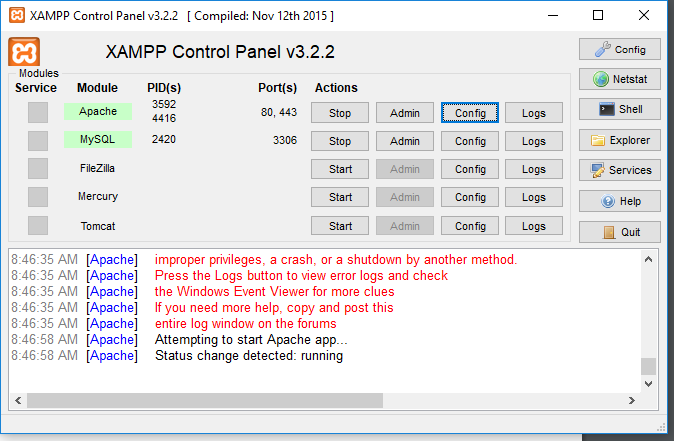


1. Enter '127.0.0.1 && type index.php' and take snapshot

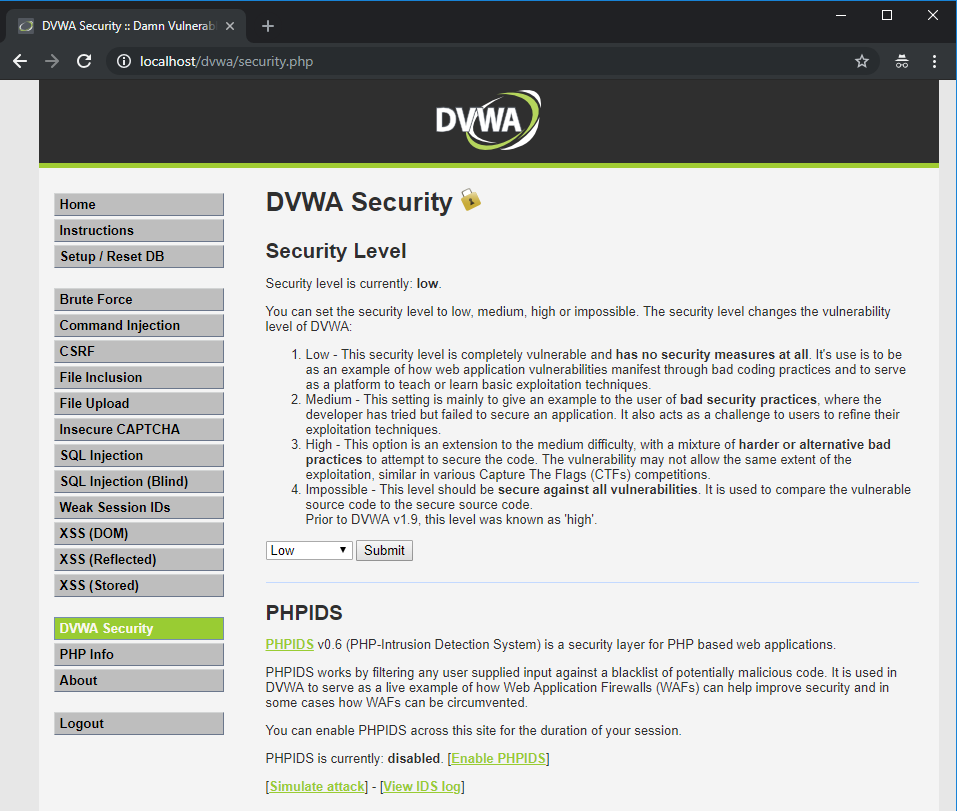


# Pre-requisite step

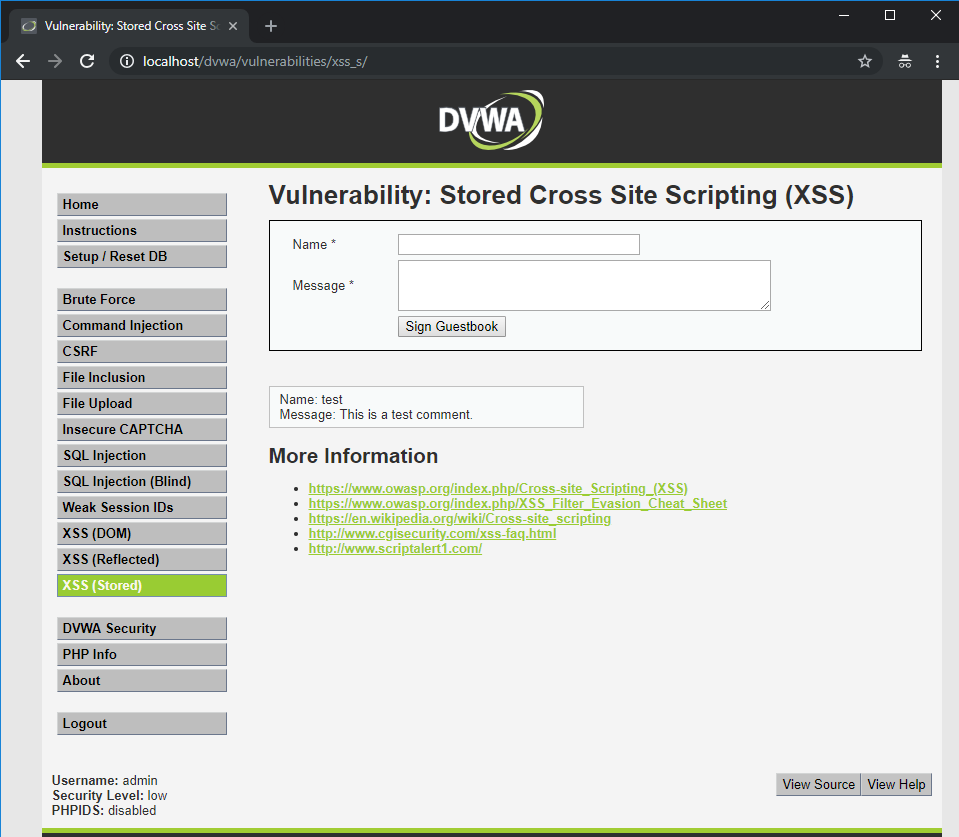
## Setup XAMPP+DVWA (as before) (Take snapshot)



## Login DVWA and Set Security Low (Take snapshot)



## Goto 'XSS Stored' (Take snapshot)

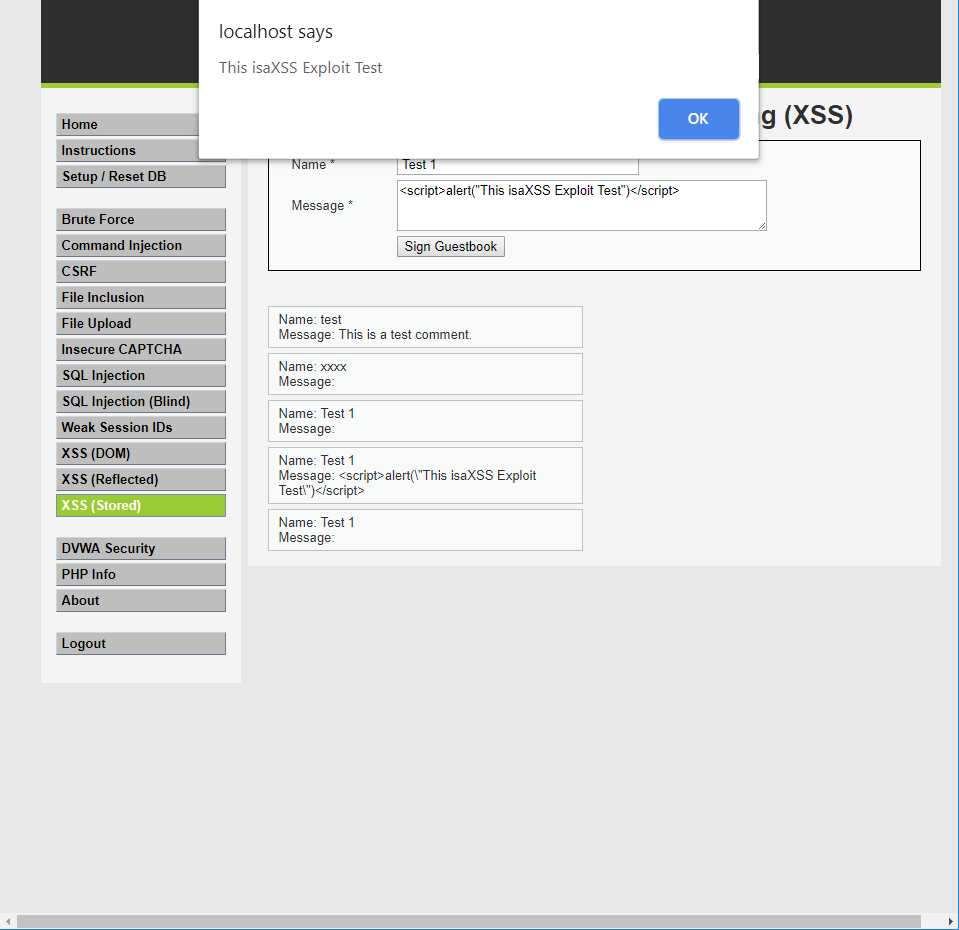


## A) Basic XSS Test

1. Name: Test 1

2. Message: <script>alert("This is a XSS Exploit Test")</script>

3. Click Sign Guestbook

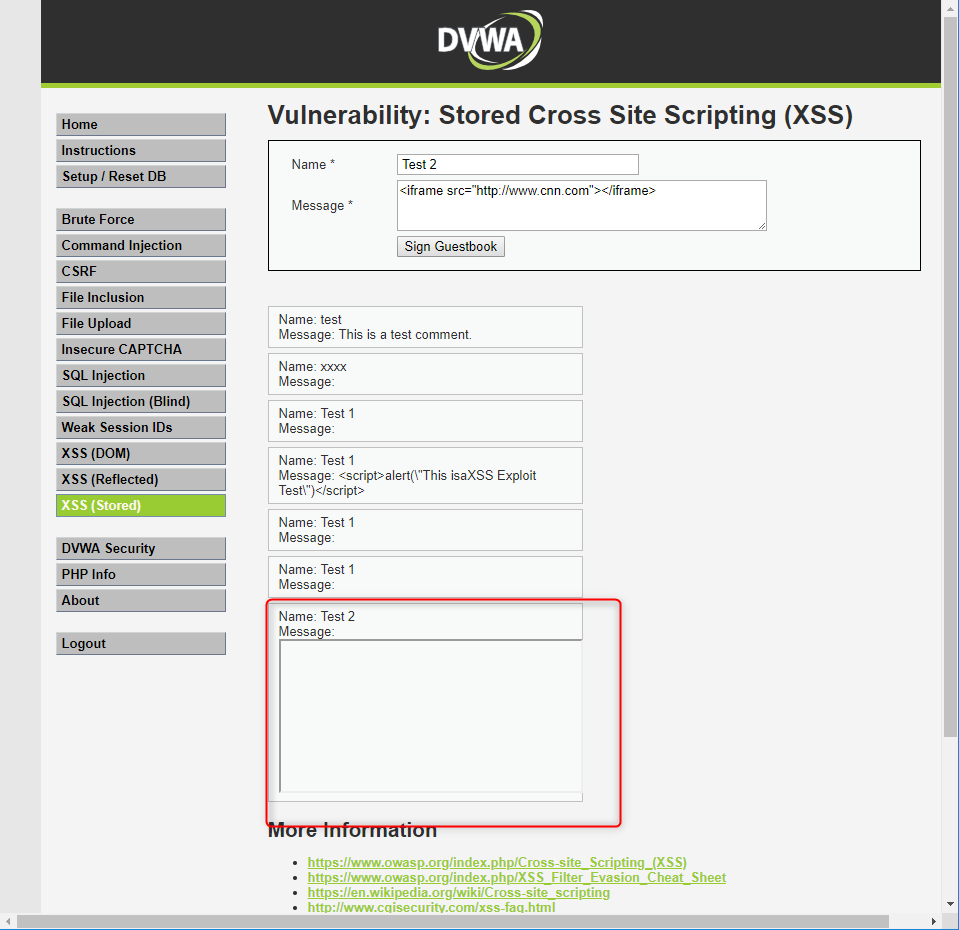


## B) XSS Test 2 (Take snapshot)

0. Name: Test 2

1. Message: <iframe src="http://www.cnn.com"></iframe>

2. Click Sign Guestbook

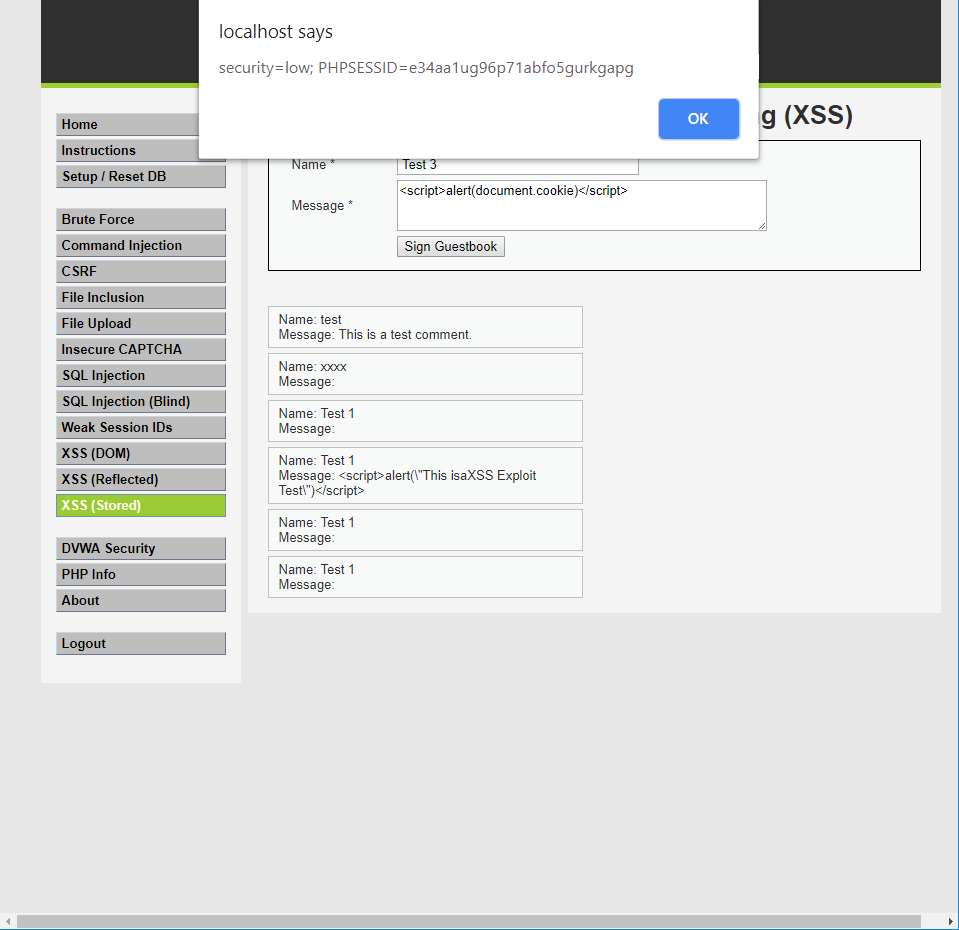


## C) XSS Test 3 (Take snapshot)

0. Name: Test 3

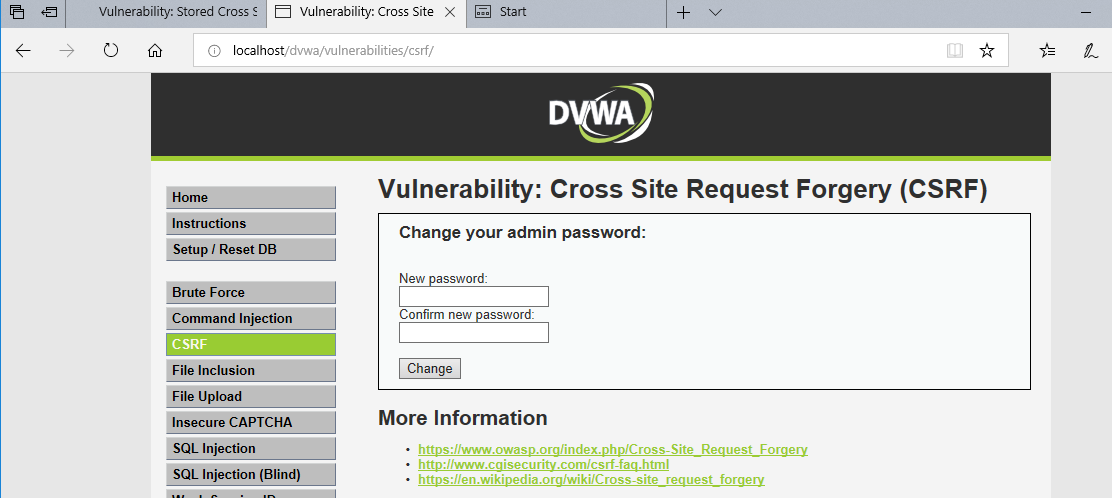
1. Message: <script>alert(document.cookie)</script>

2. Click Sign Guestbook



# CSRF

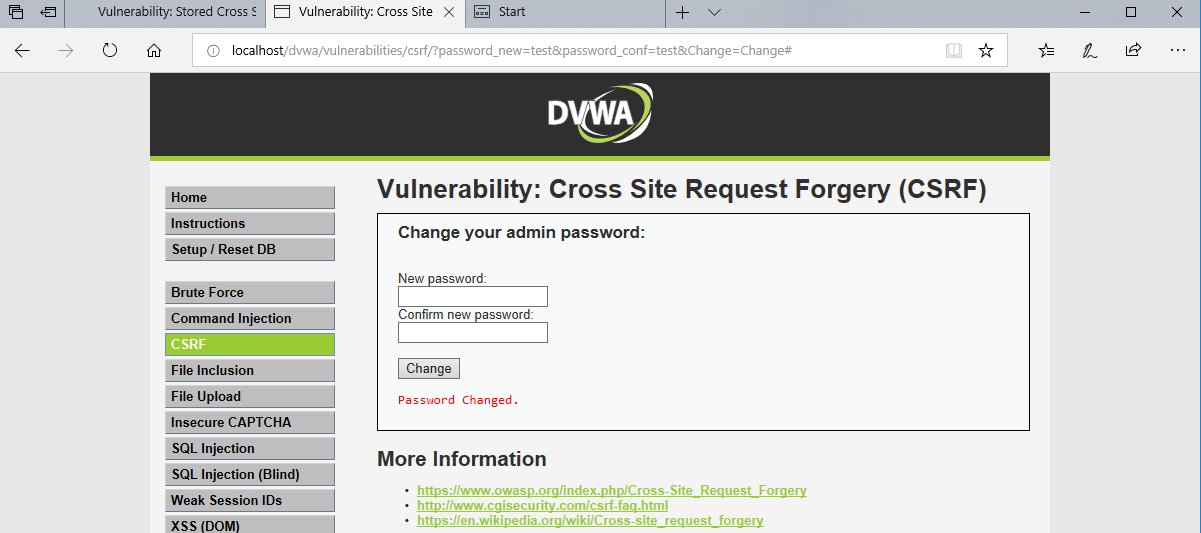
## Step 1: In the DVWA navigation menu, click the CSRF button



## Step 2: As a valid user of the Web application, type boxes on the following data into the CSRF page and click Change to change the admin password. (Take snapshot)

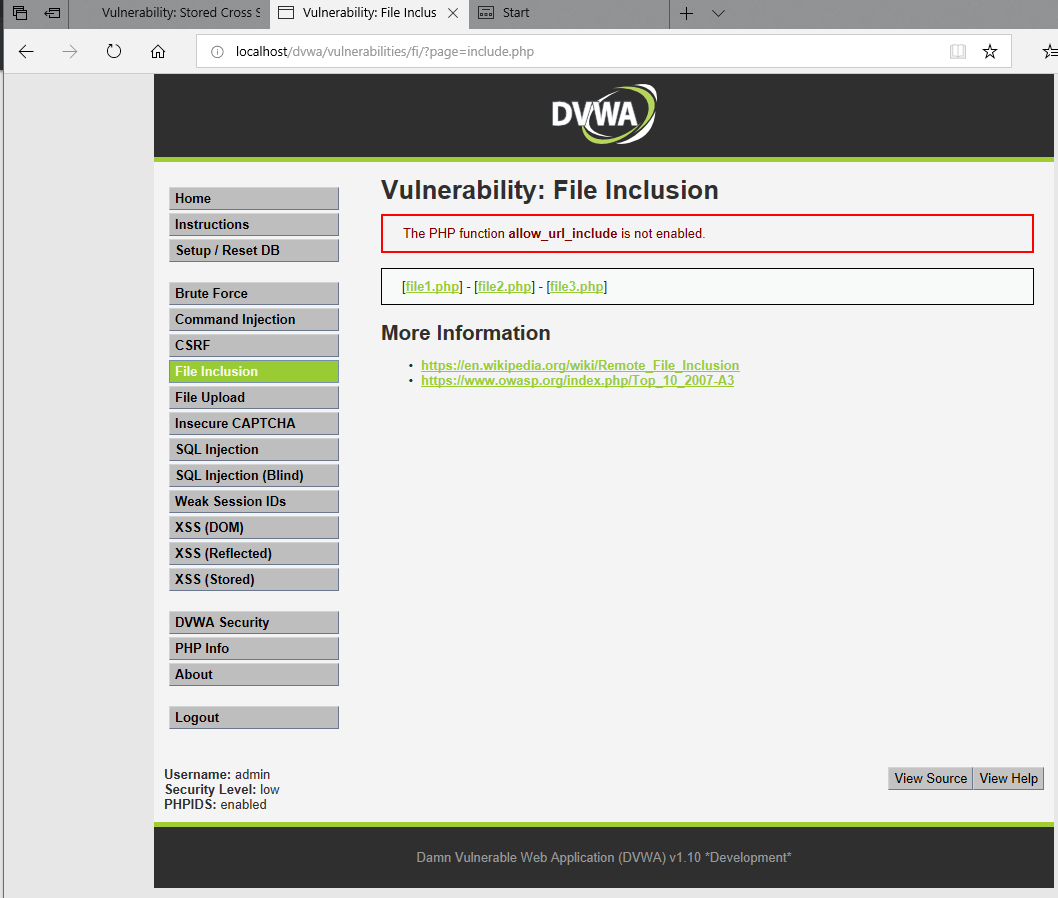
New password: **test**

Confirm new password: **test**



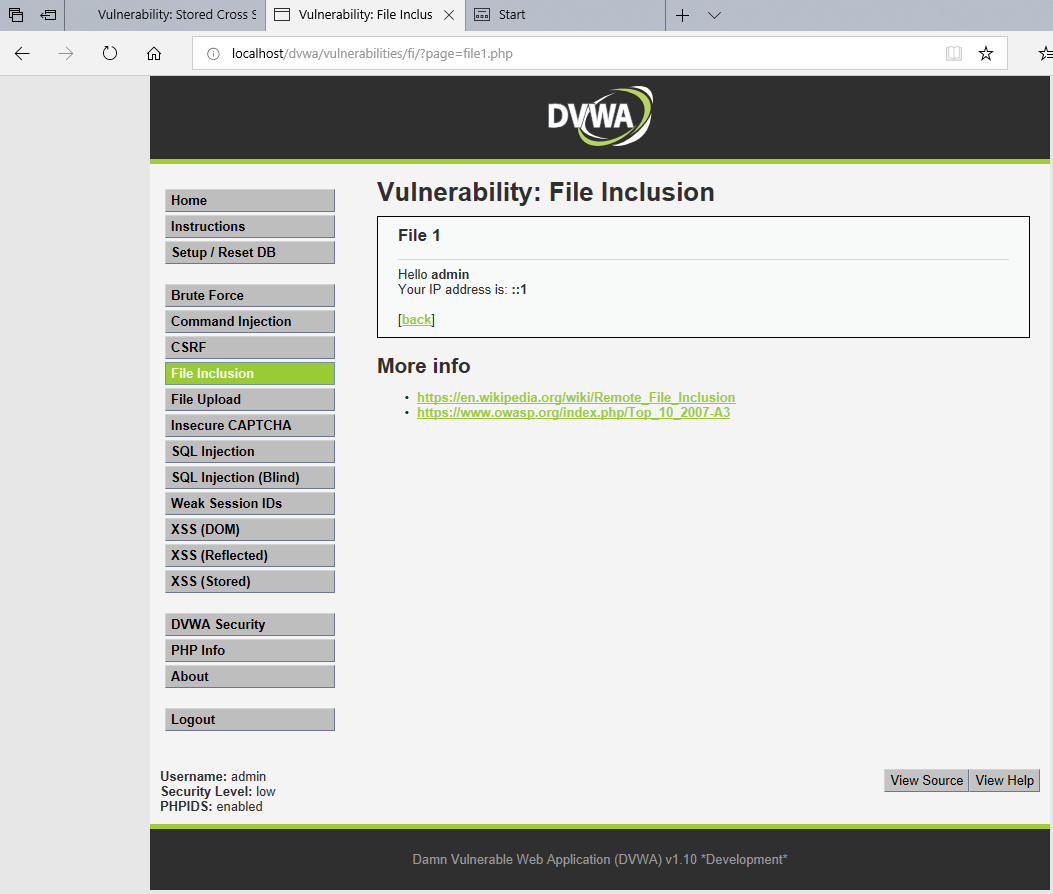
# File Inclusion

## Step 1: In the DVWA navigation menu, click the File Inclusion button.

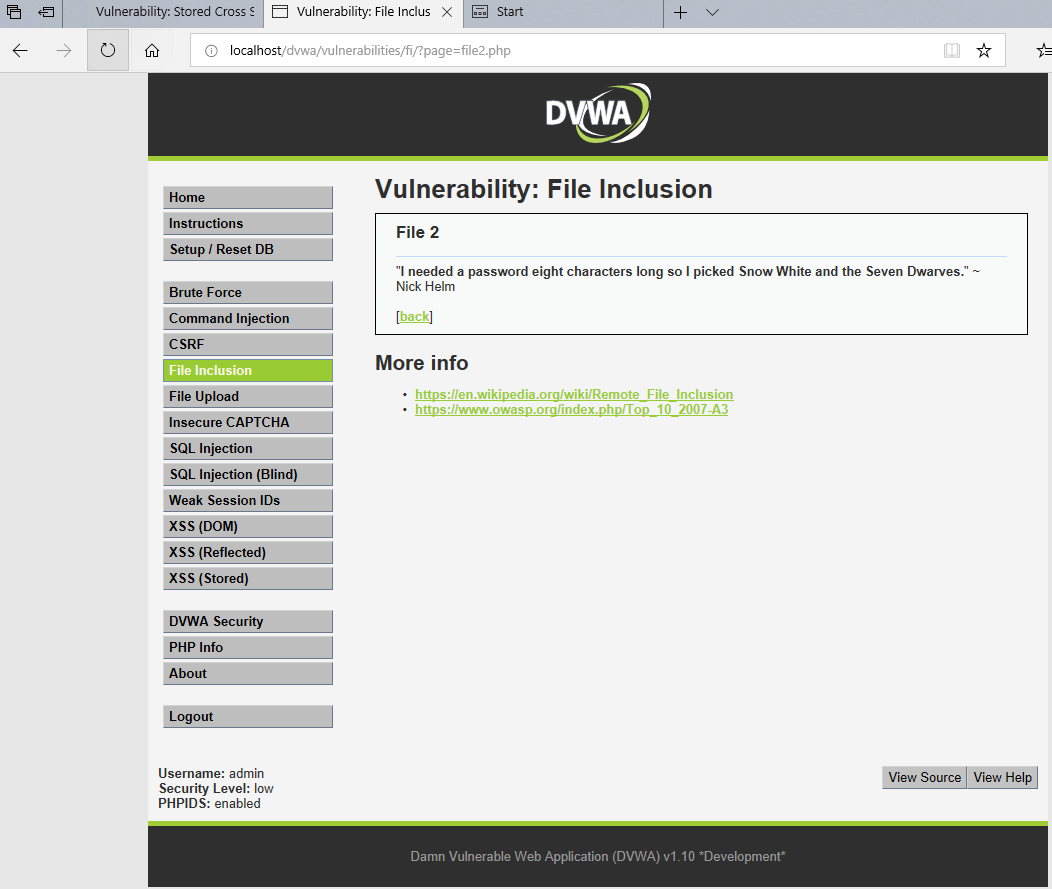


## Step 2: (Take snapshot)

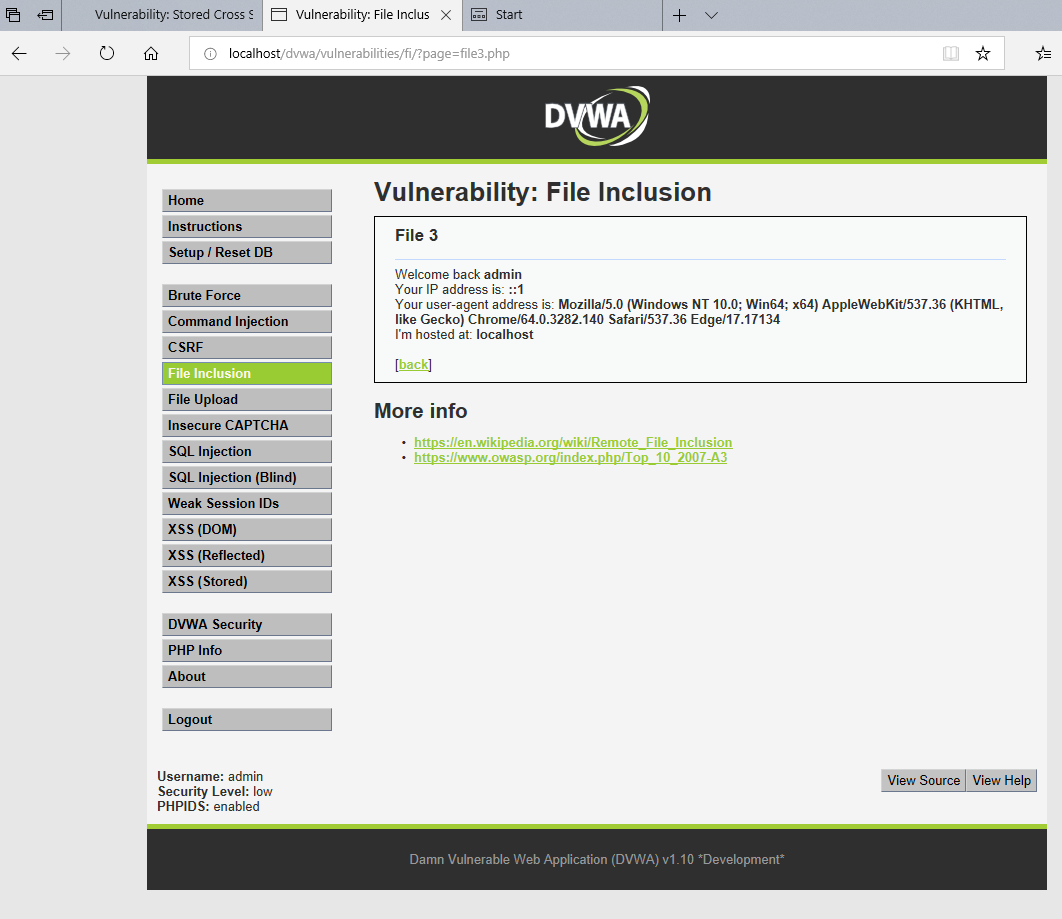
In the URL, highlight include.php and Type file1.php, so that the complete URL reads: http://172.30.0.11/dvwa/vulnerabilities/fi/?page=file1.php, and press Enter to submit the script.



Show file 2:



Show file 3:



Show file 4:

