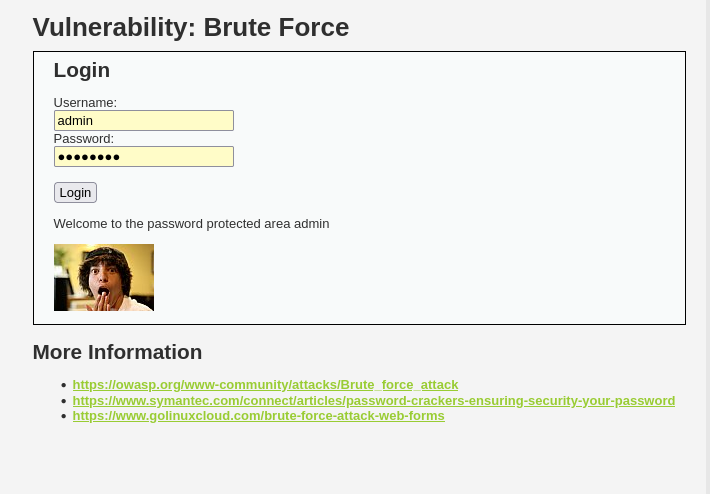
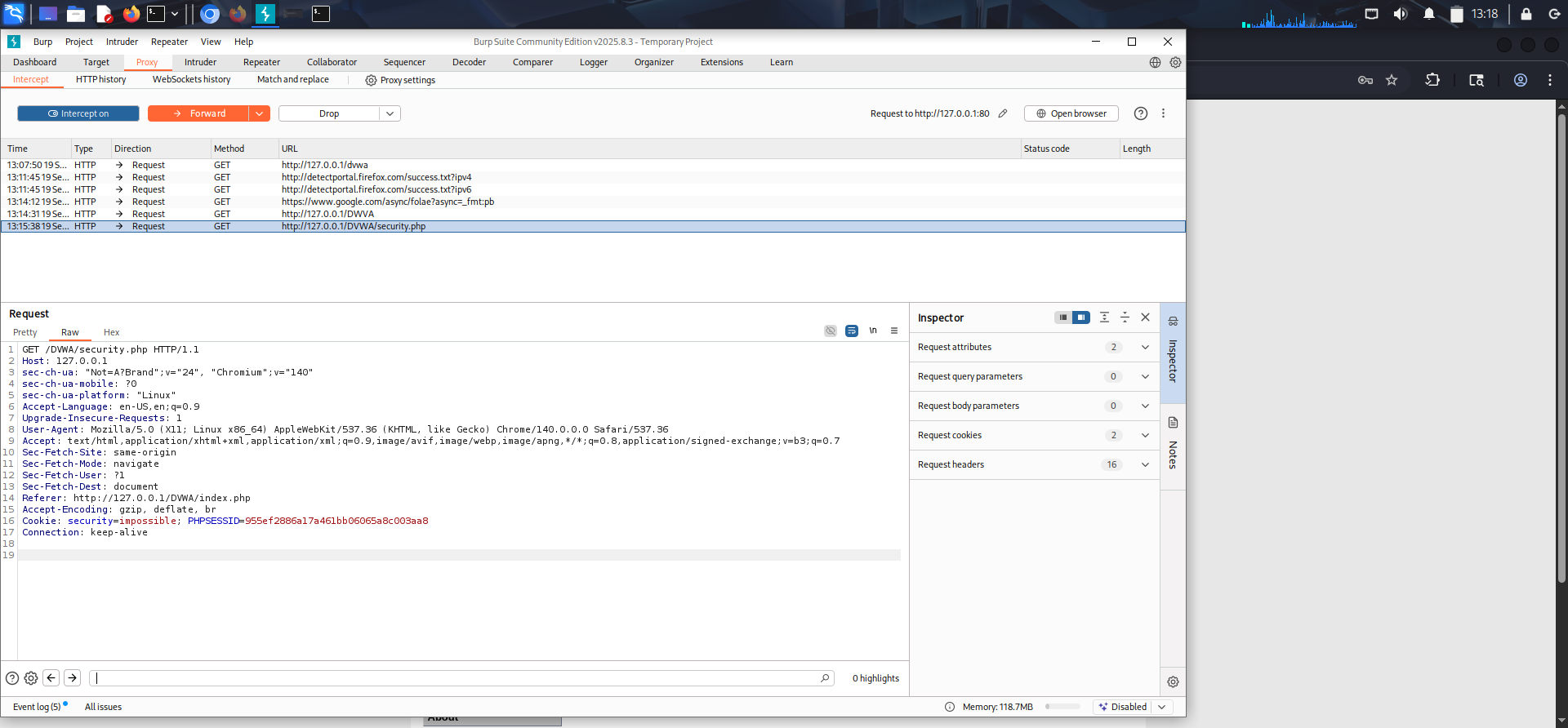
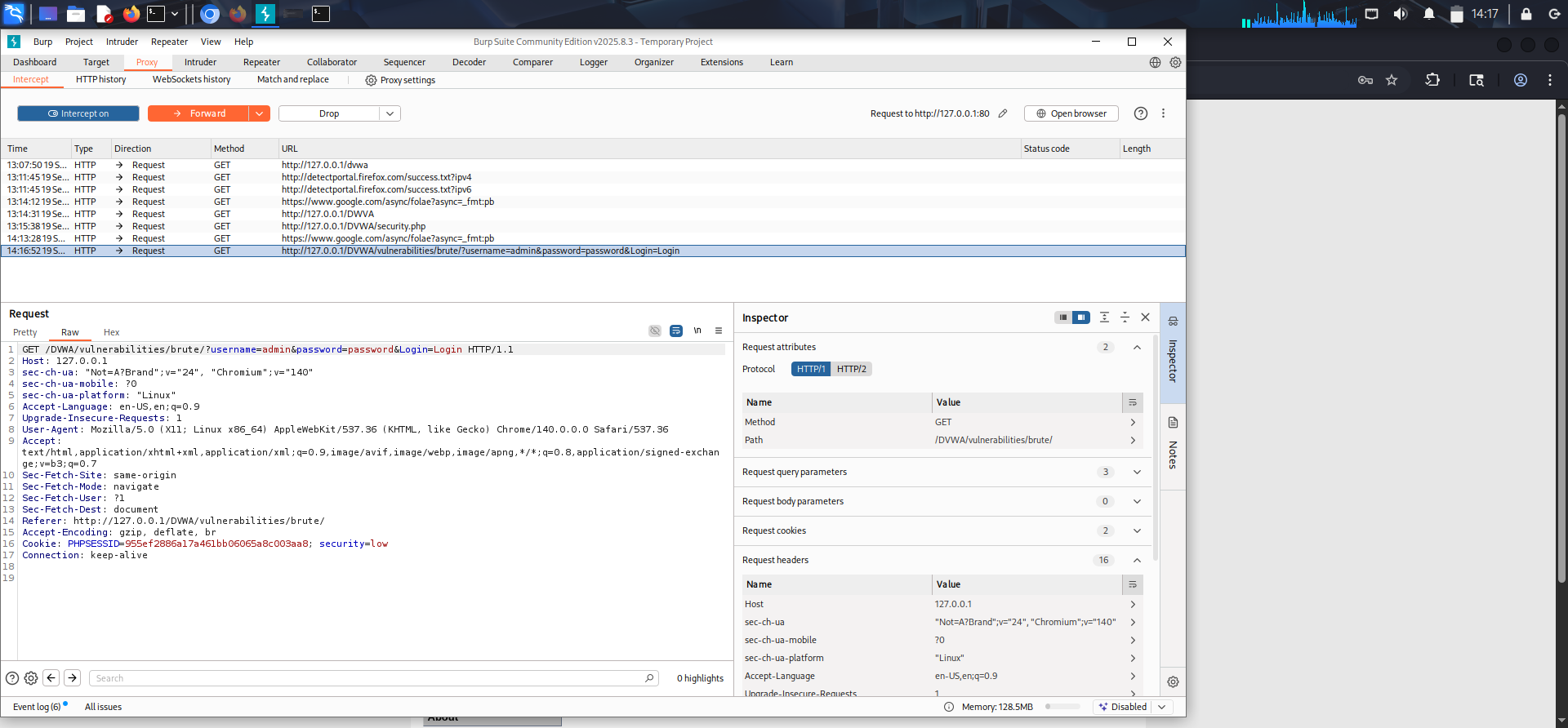
**LAB EXERCISE 8**

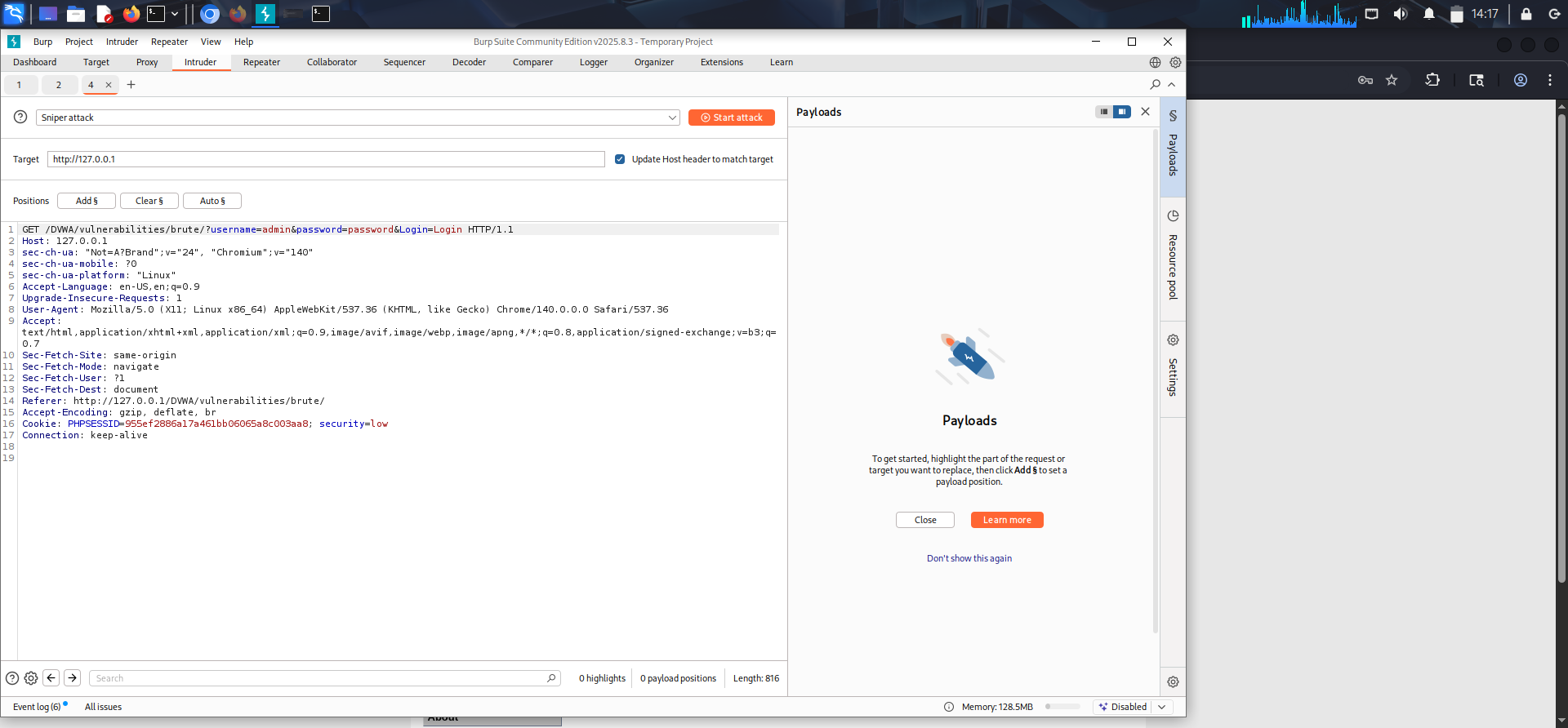
**Exercise 8.1:** Perform Brute force attack and Dictionary Attack on the login page of the vulnerable application to identify users and their passwords.

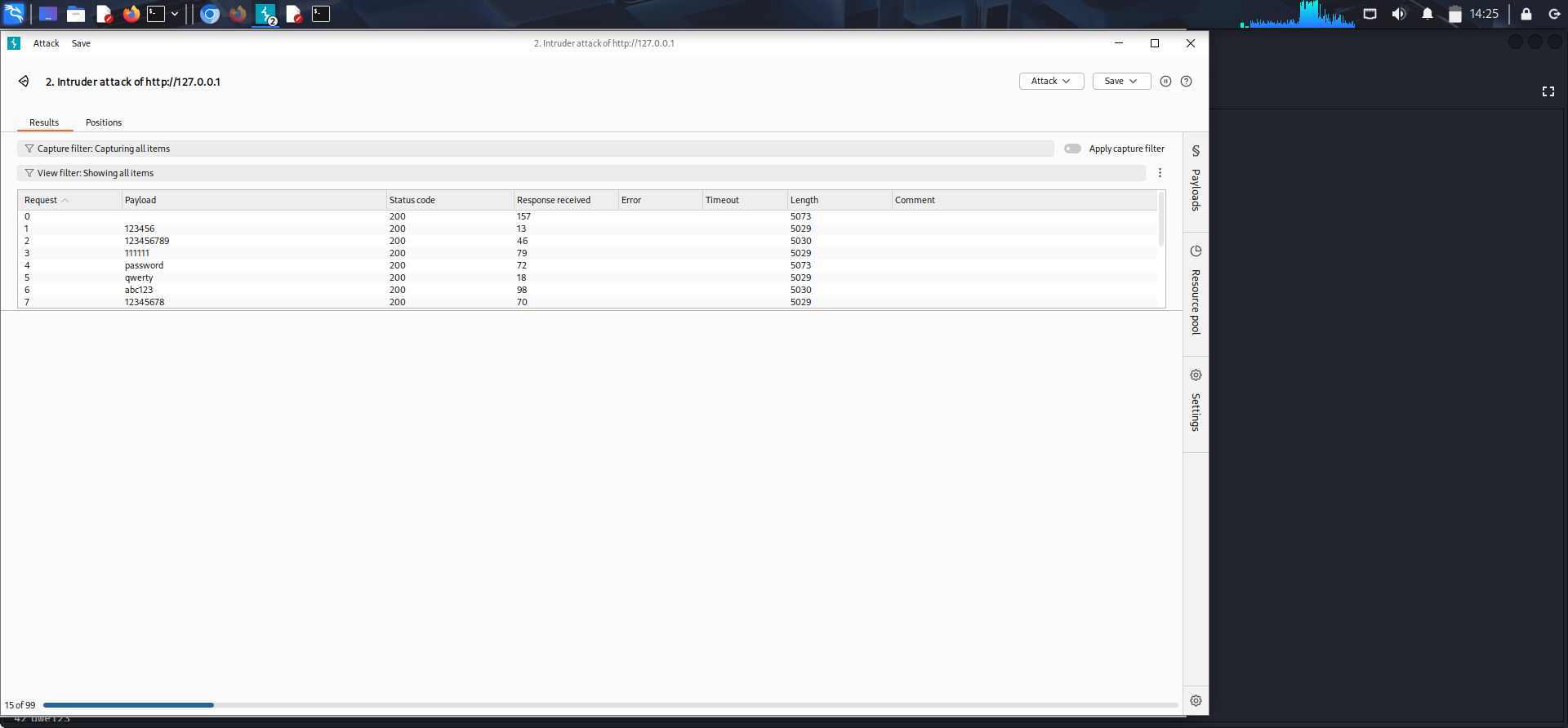


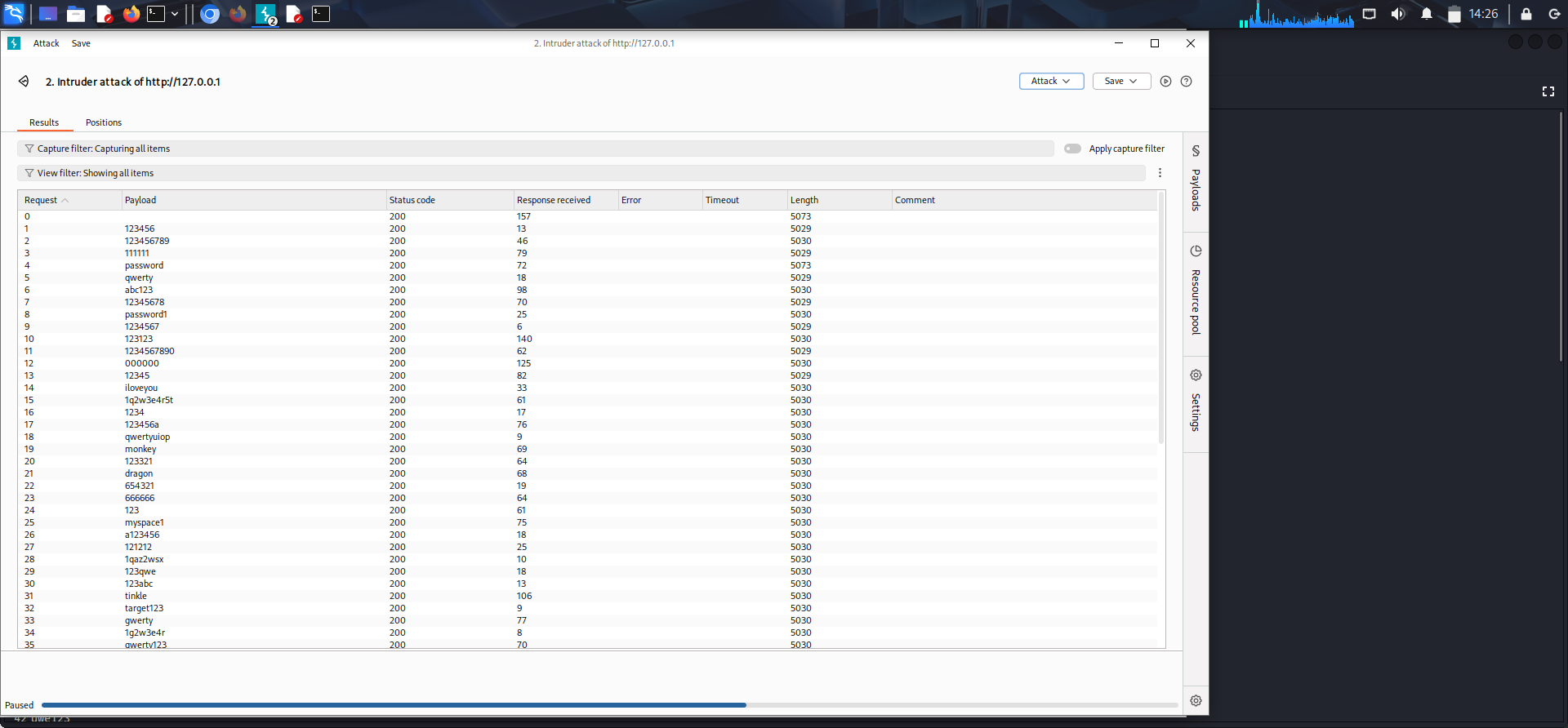




Setting payloads

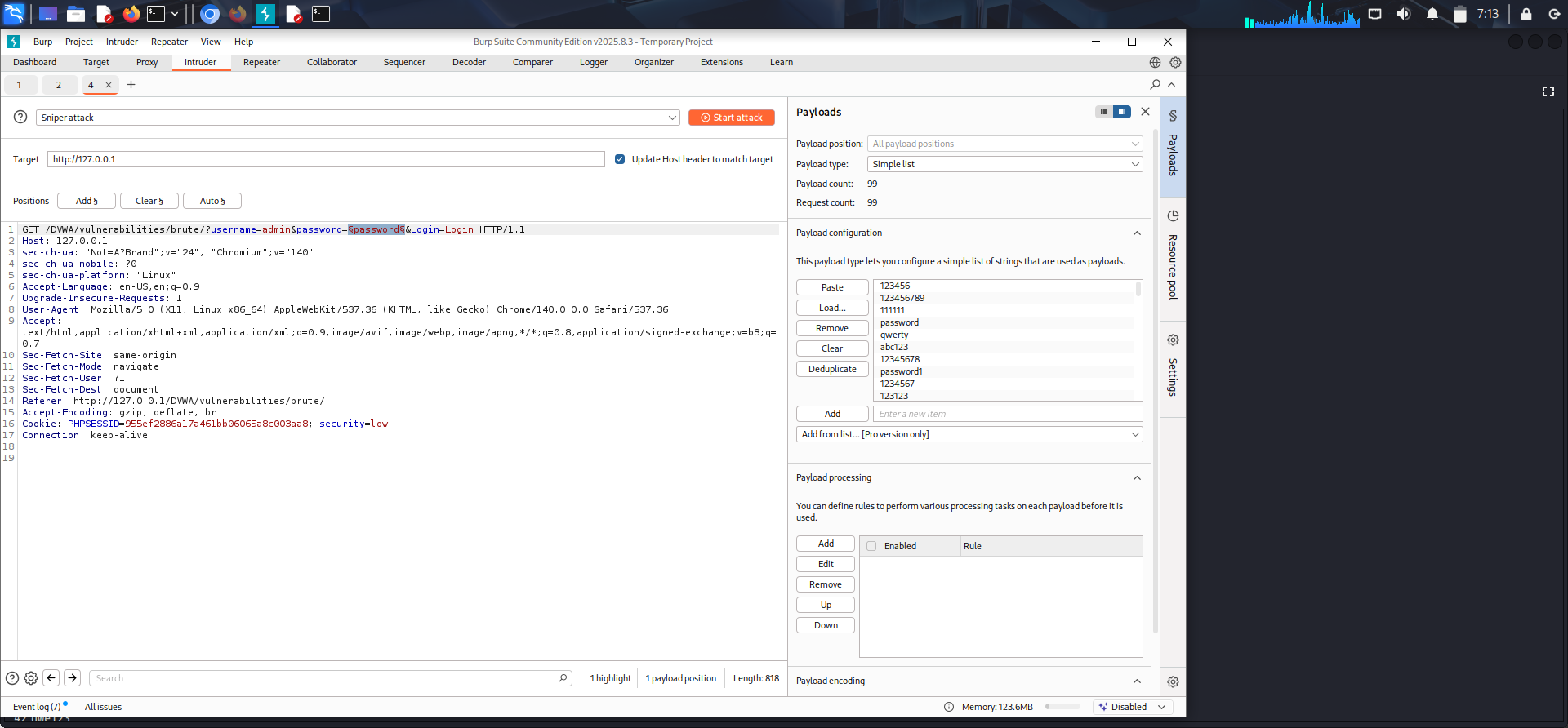






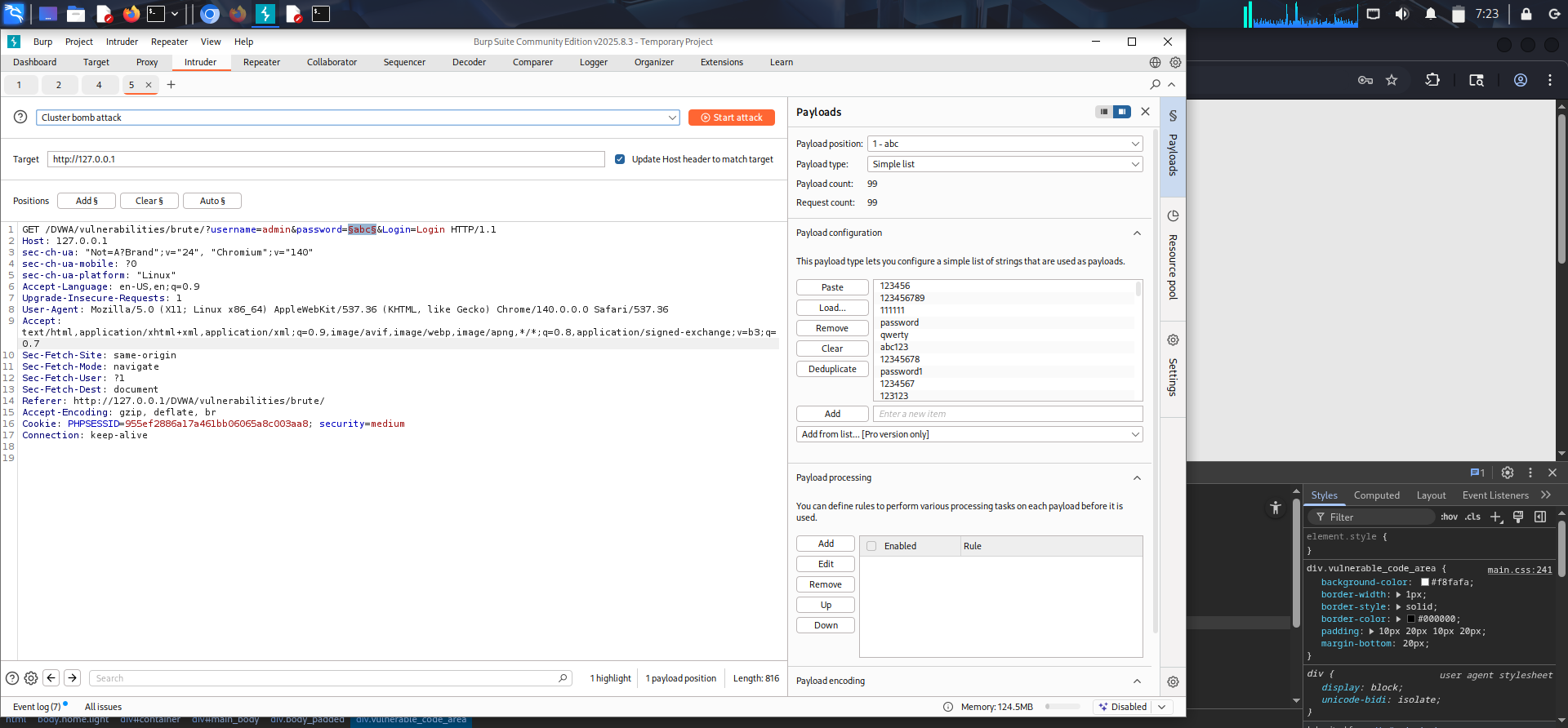
It shows that valid username and password combination is admin, password.

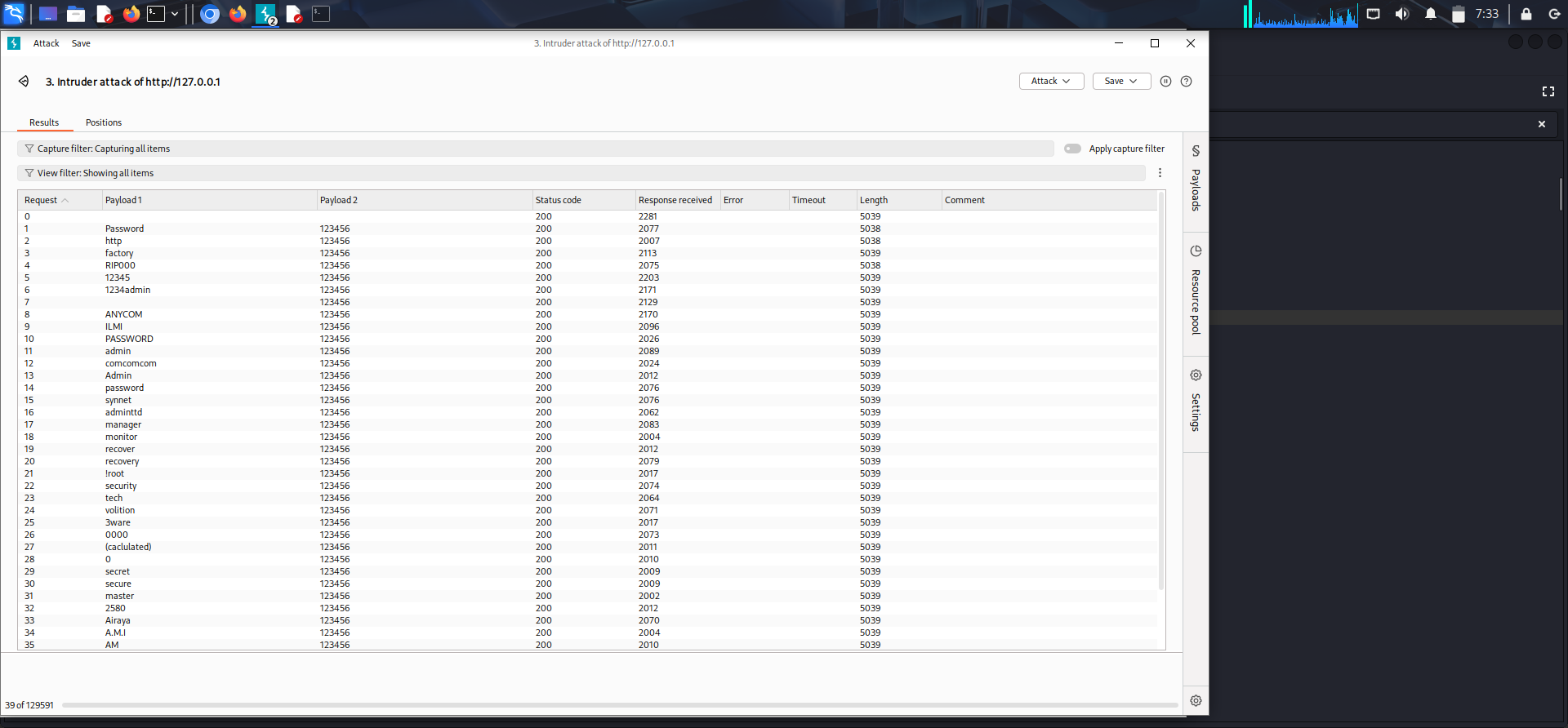
We are using burpsuite for dictionary attacks on dvwa security low level.



**Medium Level Security**

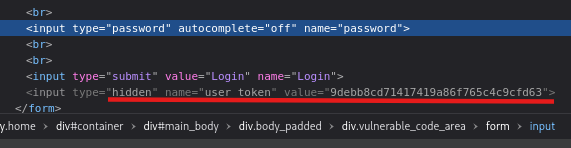


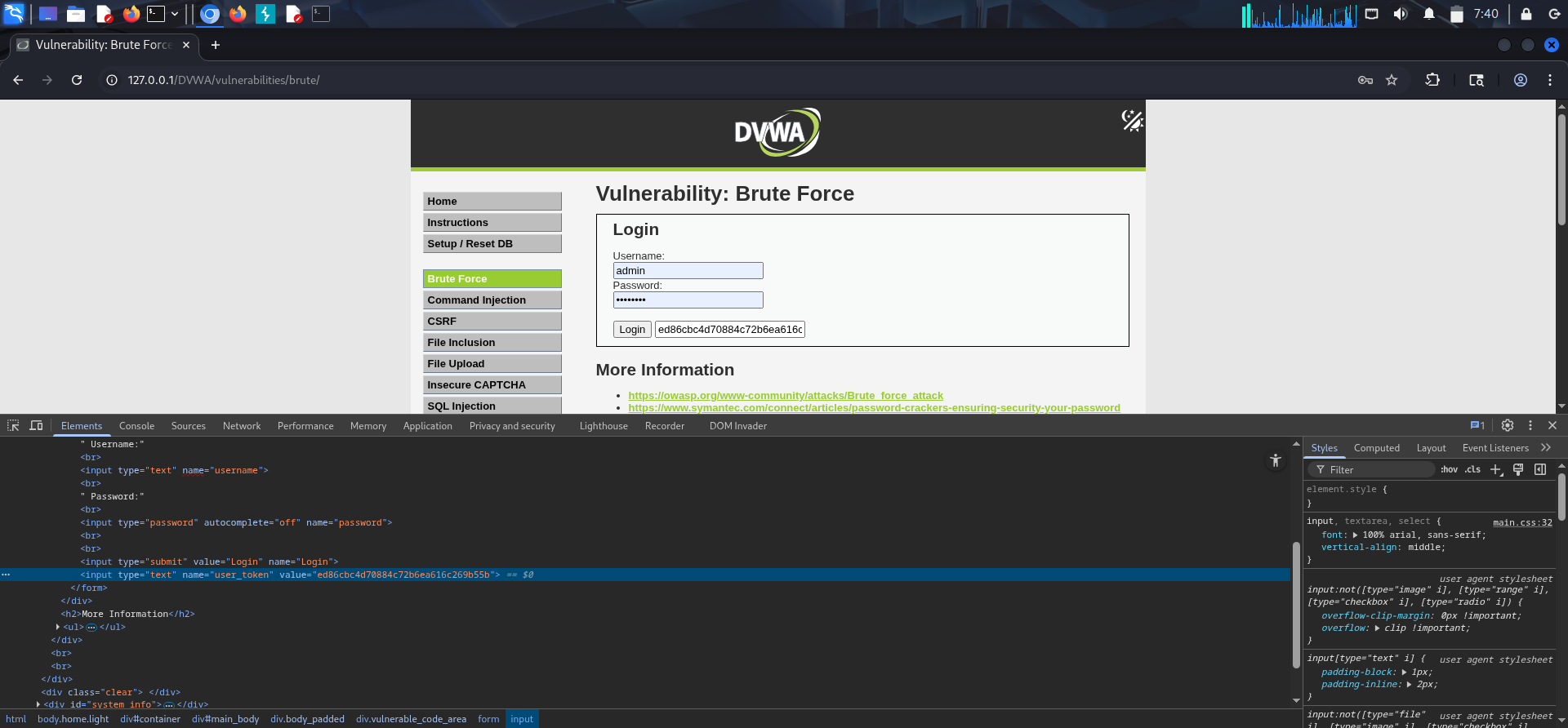


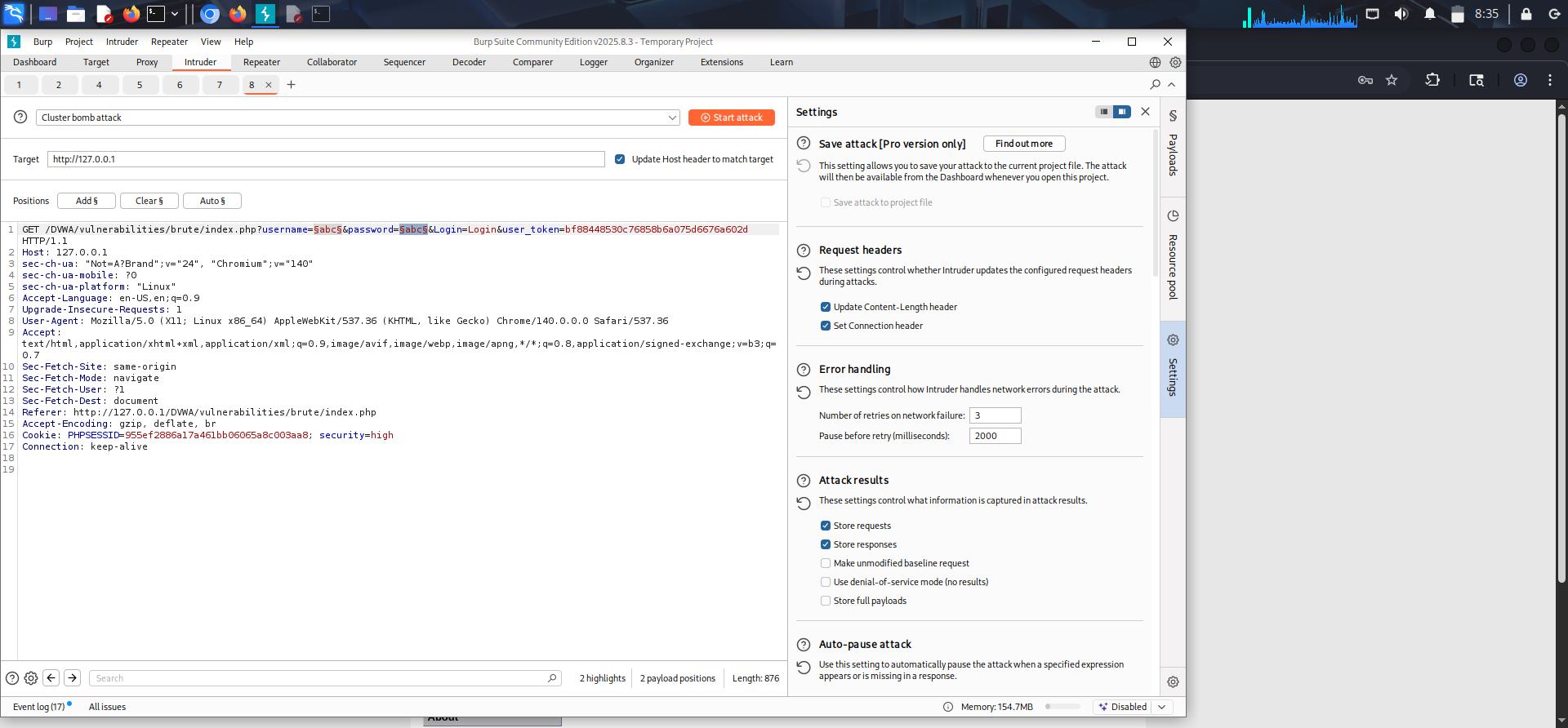


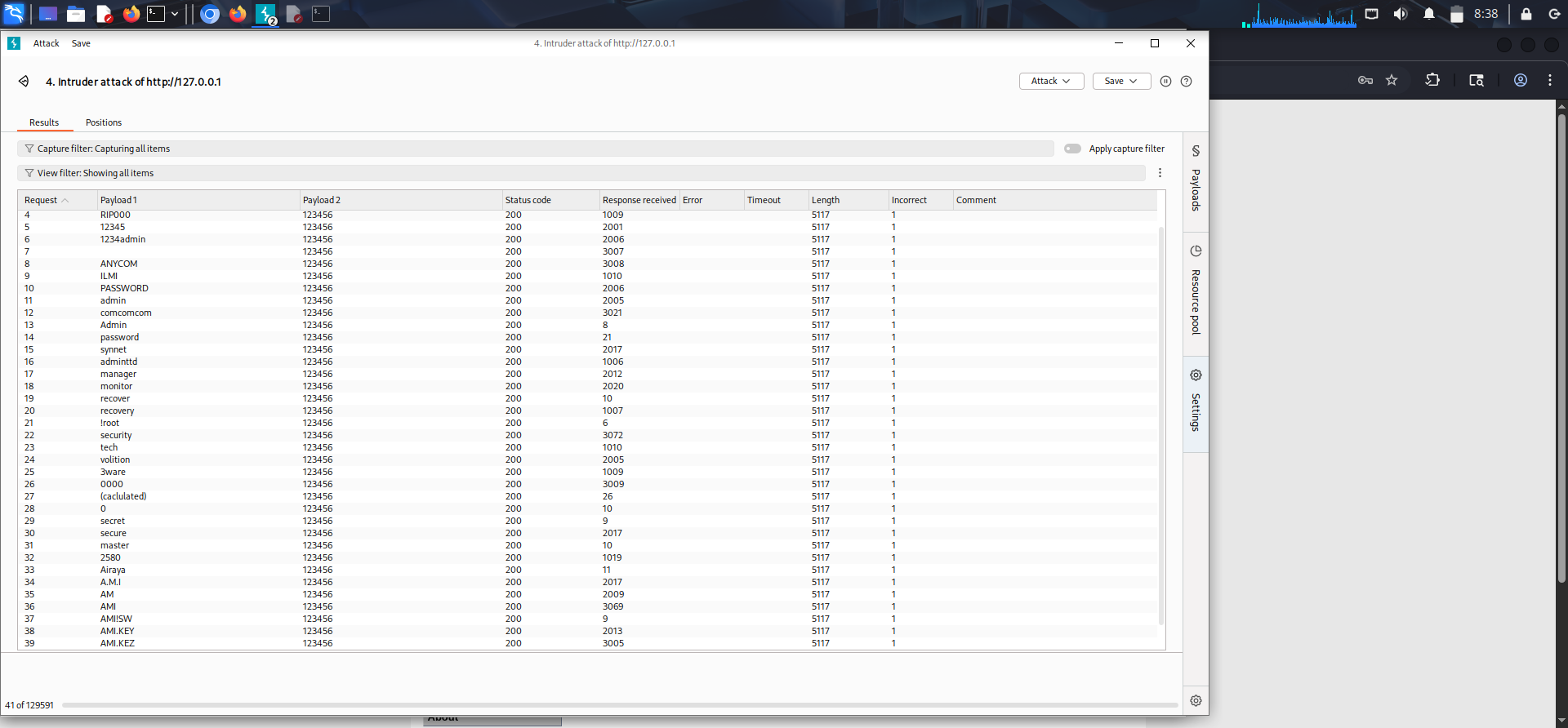
**High Level Security**







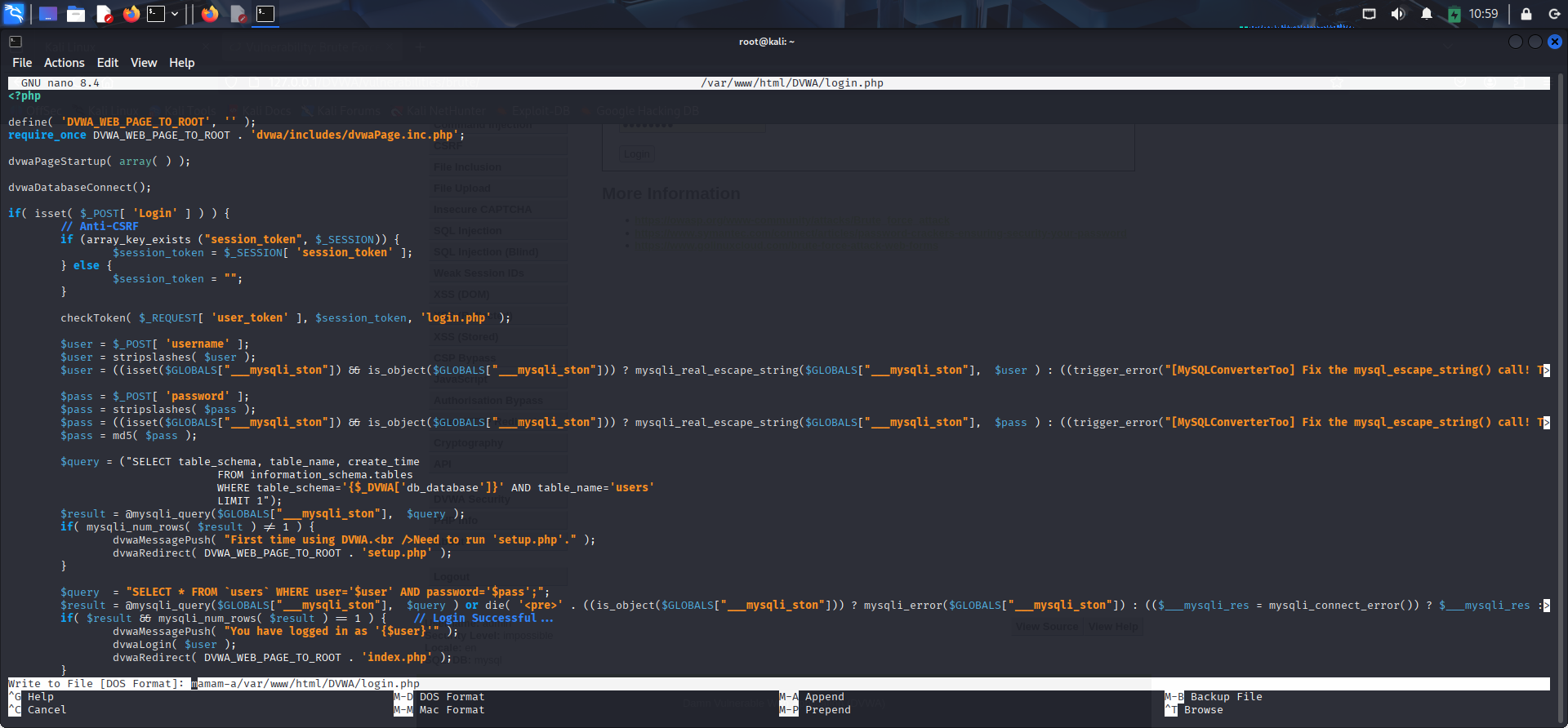




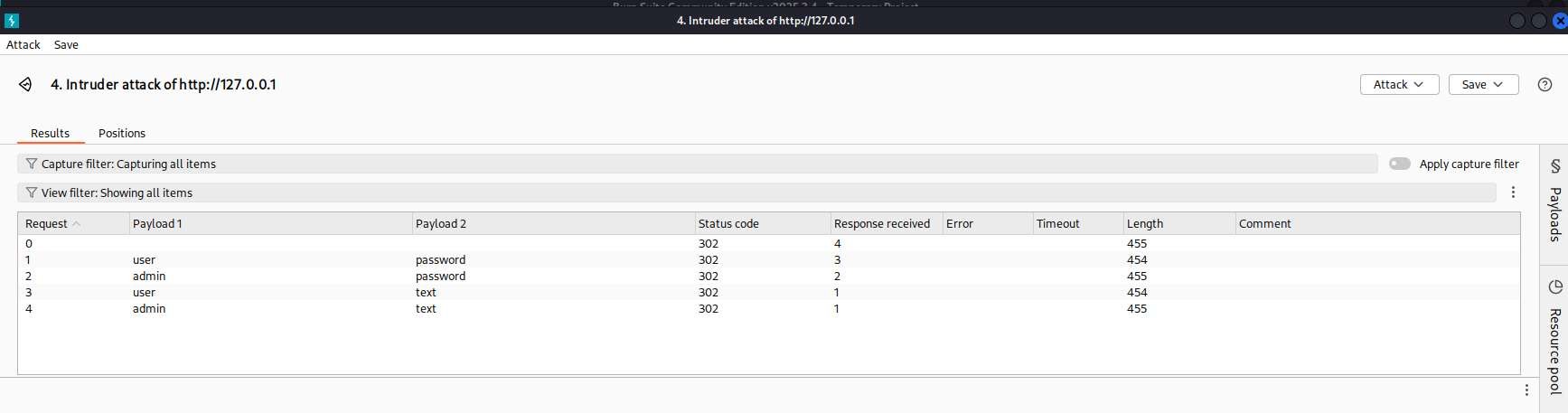
1. Anti CSRF token protected code

|  |
| --- |
| <?php  if( isset( $\_POST[ 'Login' ] ) && isset ($\_POST['username']) && isset ($\_POST['password']) ) {  // Check Anti-CSRF token  checkToken( $\_REQUEST[ 'user\_token' ], $\_SESSION[ 'session\_token' ], 'index.php' );  // Sanitise username input  $user = $\_POST[ 'username' ];  $user = stripslashes( $user );  $user = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], $user ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));  // Sanitise password input  $pass = $\_POST[ 'password' ];  $pass = stripslashes( $pass );  $pass = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], $pass ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));  $pass = md5( $pass );  // Default values  $total\_failed\_login = 3;  $lockout\_time = 15;  $account\_locked = false;  // Check the database (Check user information)  $data = $db->prepare( 'SELECT failed\_login, last\_login FROM users WHERE user = (:user) LIMIT 1;' );  $data->bindParam( ':user', $user, PDO::PARAM\_STR );  $data->execute();  $row = $data->fetch();  // Check to see if the user has been locked out.  if( ( $data->rowCount() == 1 ) && ( $row[ 'failed\_login' ] >= $total\_failed\_login ) ) {  // User locked out. Note, using this method would allow for user enumeration!  //echo "<pre><br />This account has been locked due to too many incorrect logins.</pre>";  // Calculate when the user would be allowed to login again  $last\_login = strtotime( $row[ 'last\_login' ] );  $timeout = $last\_login + ($lockout\_time \* 60);  $timenow = time();  /\*  print "The last login was: " . date ("h:i:s", $last\_login) . "<br />";  print "The timenow is: " . date ("h:i:s", $timenow) . "<br />";  print "The timeout is: " . date ("h:i:s", $timeout) . "<br />";  \*/  // Check to see if enough time has passed, if it hasn't locked the account  if( $timenow < $timeout ) {  $account\_locked = true;  // print "The account is locked<br />";  }  }  // Check the database (if username matches the password)  $data = $db->prepare( 'SELECT \* FROM users WHERE user = (:user) AND password = (:password) LIMIT 1;' );  $data->bindParam( ':user', $user, PDO::PARAM\_STR);  $data->bindParam( ':password', $pass, PDO::PARAM\_STR );  $data->execute();  $row = $data->fetch();  // If its a valid login...  if( ( $data->rowCount() == 1 ) && ( $account\_locked == false ) ) {  // Get users details  $avatar = $row[ 'avatar' ];  $failed\_login = $row[ 'failed\_login' ];  $last\_login = $row[ 'last\_login' ];  // Login successful  echo "<p>Welcome to the password protected area <em>{$user}</em></p>";  echo "<img src=\"{$avatar}\" />";  // Had the account been locked out since last login?  if( $failed\_login >= $total\_failed\_login ) {  echo "<p><em>Warning</em>: Someone might of been brute forcing your account.</p>";  echo "<p>Number of login attempts: <em>{$failed\_login}</em>.<br />Last login attempt was at: <em>{$last\_login}</em>.</p>";  }  // Reset bad login count  $data = $db->prepare( 'UPDATE users SET failed\_login = "0" WHERE user = (:user) LIMIT 1;' );  $data->bindParam( ':user', $user, PDO::PARAM\_STR );  $data->execute();  } else {  // Login failed  sleep( rand( 2, 4 ) );  // Give the user some feedback  echo "<pre><br />Username and/or password incorrect.<br /><br/>Alternative, the account has been locked because of too many failed logins.<br />If this is the case, <em>please try again in {$lockout\_time} minutes</em>.</pre>";  // Update bad login count  $data = $db->prepare( 'UPDATE users SET failed\_login = (failed\_login + 1) WHERE user = (:user) LIMIT 1;' );  $data->bindParam( ':user', $user, PDO::PARAM\_STR );  $data->execute();  }  // Set the last login time  $data = $db->prepare( 'UPDATE users SET last\_login = now() WHERE user = (:user) LIMIT 1;' );  $data->bindParam( ':user', $user, PDO::PARAM\_STR );  $data->execute();  }  // Generate Anti-CSRF token  generateSessionToken();  ?> |

**Added Security Code:**



**Testing and Validation:**

Top of Form

Bottom of Form