

Practical 6

Aim: Practical on Exploiting Web-based applications

1. Reconnaissance and Identification of Web applications

Run the following commands to make sure that you Kali Linux distribution is up to date.

- sudo apt update
- sudo apt upgrade
- sudo apt dist-upgrade

The process of waf detection can be automated using nmap script http-waf-detect.nse as shown below:

```

kali@kali:~$ sudo apt-get update
[sudo] password for kali:
Hit:1 https://packages.sury.org/php buster InRelease
Get:2 https://dl.google.com/linux/chrome/deb stable InRelease [1,825 B]
Get:3 http://kali.download/kali kali-rolling InRelease [41.2 kB]
Get:4 http://kali.download/kali kali-rolling/main amd64 Packages [19.4 MB]
Get:5 https://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1,082 B]
Get:6 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [45.6 MB]
Get:7 http://kali.download/kali kali-rolling/contrib amd64 Packages [121 kB]
Get:8 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [294 kB]
Get:9 http://kali.download/kali kali-rolling/non-free amd64 Packages [226 kB]
Get:10 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [914 kB]
Fetched 66.6 MB in 17s (3,912 kB/s)
Reading package lists... Done

kali@kali:~$
kali@kali:~$ sudo nmap -vv -p 80 --script http-waf-detect www.testfire.net
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-14 01:04 EST
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 01:04
Completed NSE at 01:04, 0.00s elapsed
Initiating Ping Scan at 01:04
Scanning www.testfire.net (65.61.137.117) [4 ports]
Completed Ping Scan at 01:04, 0.02s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 01:04
Completed Parallel DNS resolution of 1 host. at 01:04, 0.36s elapsed
Initiating SYN Stealth Scan at 01:04
Scanning www.testfire.net (65.61.137.117) [1 port]
Completed SYN Stealth Scan at 01:04, 0.23s elapsed (1 total ports)
NSE: Script scanning 65.61.137.117.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 01:04
Completed NSE at 01:04, 0.00s elapsed
Nmap scan report for www.testfire.net (65.61.137.117)
Host is up, received reset ttl 128 (0.0012s latency).
Scanned at 2023-12-14 01:04:39 EST for 0s

PORT      STATE      SERVICE REASON
80/tcp    filtered  http    no-response

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 01:04
Completed NSE at 01:04, 0.00s elapsed
Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.90 seconds
Raw packets sent: 6 (240B) | Rcvd: 1 (40B)

```

Then we will run the python tool WAFW00F to perform the identification and fingerprinting of a Web Application Firewall. In this case we will check the firewall of www.hdfcbank.com.

[illegible]

Then we will use a Load Balancing Detector on www.hdfcbank.com.

```
(kali@kali)-[~]
$ sudo lbd www.hdfcbank.com

lbd - load balancing detector 0.4 - Checks if a given domain uses load-balancing.
    Written by Stefan Behte (http://ge.mine.nu)
    Proof-of-concept! Might give false positives.

Checking for DNS-Loadbalancing: FOUND
www.hdfcbank.com has address 104.17.6.56
www.hdfcbank.com has address 104.16.36.67

Checking for HTTP-Loadbalancing [Server]:
cloudflare
NOT FOUND

Checking for HTTP-Loadbalancing [Date]: 06:06:20, 06:06:20, 06:06:21, 06:06:21, 06:06:21,
06:06:21, 06:06:21, 06:06:22, 06:06:22, 06:06:22, 06:06:22, 06:06:22, 06:06:22, 06:06:22,
06:06:23, 06:06:23, 06:06:23, 06:06:23, 06:06:23, 06:06:23, 06:06:23, 06:06:23, 06:06:23,

Checking for HTTP-Loadbalancing [Diff]: FOUND
< Expires: Thu, 14 Dec 2023 07:06:23 GMT
> Expires: Thu, 14 Dec 2023 07:06:24 GMT
< CF-RAY: 835442b7dbe91bd6-BOM
> CF-RAY: 835442b81d988489-BOM

www.hdfcbank.com does Load-balancing. Found via Methods: DNS HTTP[Diff]
```

After that, we will perform a WordPress scan on blogs.overandall.com to check for any WordPress vulnerabilities that we can exploit.

```
(kali@kali)-[~]
$ sudo wpscan --url blogs.overandall.com

WordPress Security Scanner by the WPScan Team
Version 3.8.25
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[+] URL: https://blogs.overandall.com/ [104.21.8.216]
[+] Started: Thu Dec 14 01:10:40 2023

Interesting Finding(s):

[+] Headers
| Interesting Entries:
| - x-powered-by: PHP/7.4.33
| - x-litespeed-cache: hit
| - platform: hostinger
| - x-turbo-charged-by: LiteSpeed
| - cf-cache-status: DYNAMIC
| - server: cloudflare
| - cf-ray: 835448837aee6ebc-BOM
| - alt-svc: h3=":443"; ma=86400
| Found By: Headers (Passive Detection)
| Confidence: 100%

[!] No Config Backups Found.

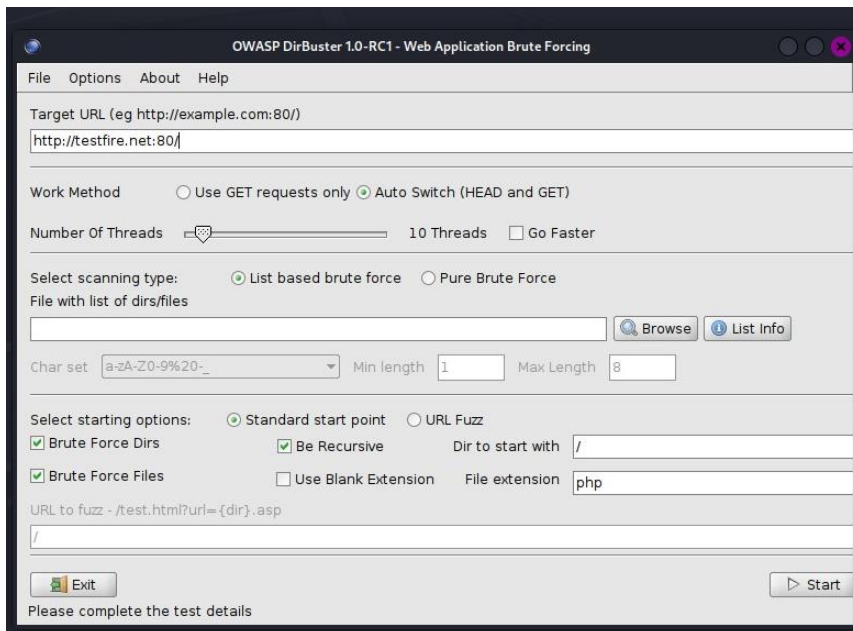
[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register

[+] Finished: Thu Dec 14 01:10:49 2023
[+] Requests Done: 139
[+] Cached Requests: 39
[+] Data Sent: 36.982 KB
[+] Data Received: 54.97 KB
[+] Memory used: 259.383 MB
[+] Elapsed time: 00:00:09

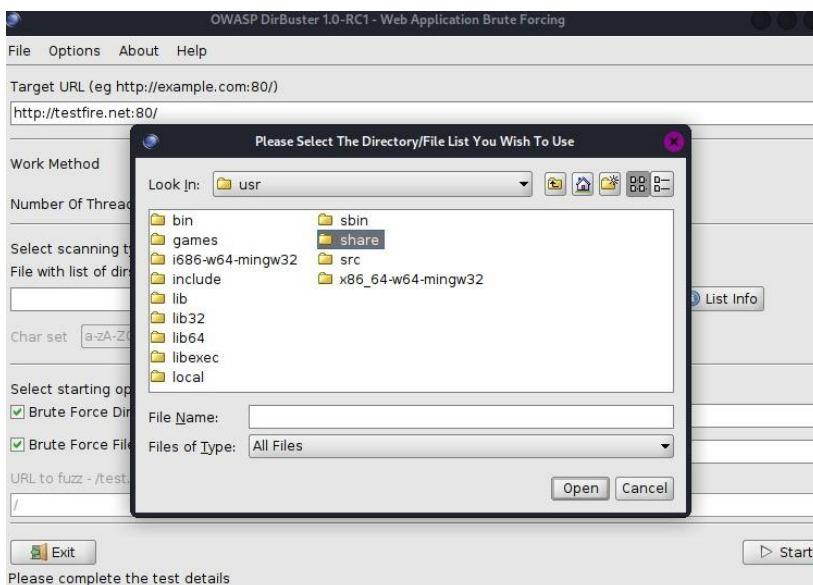
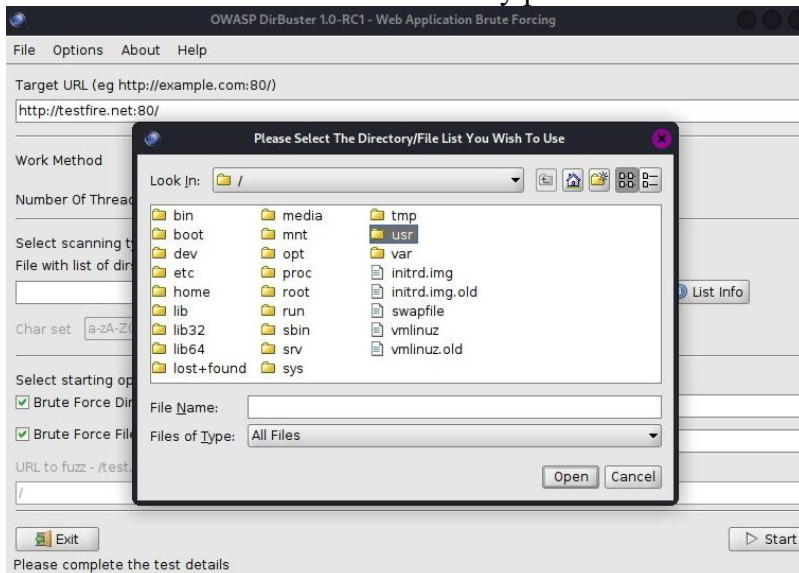
(kali@kali)-[~]
$
```

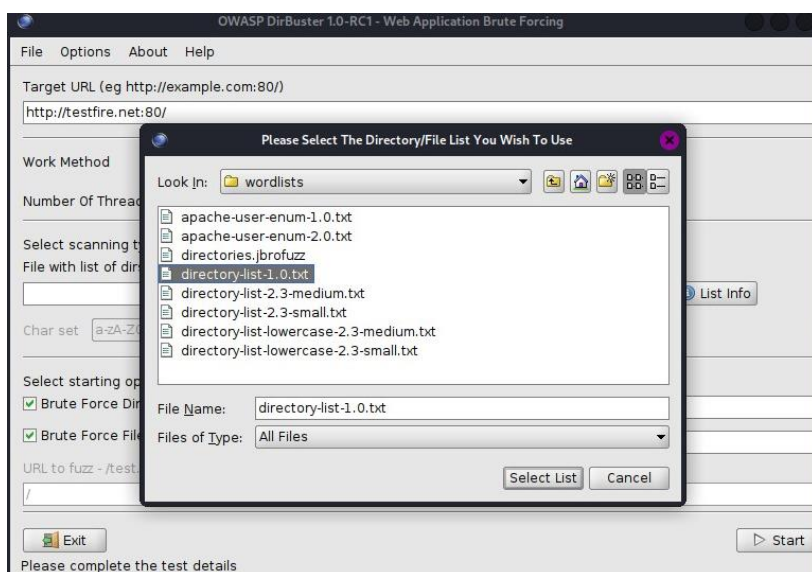
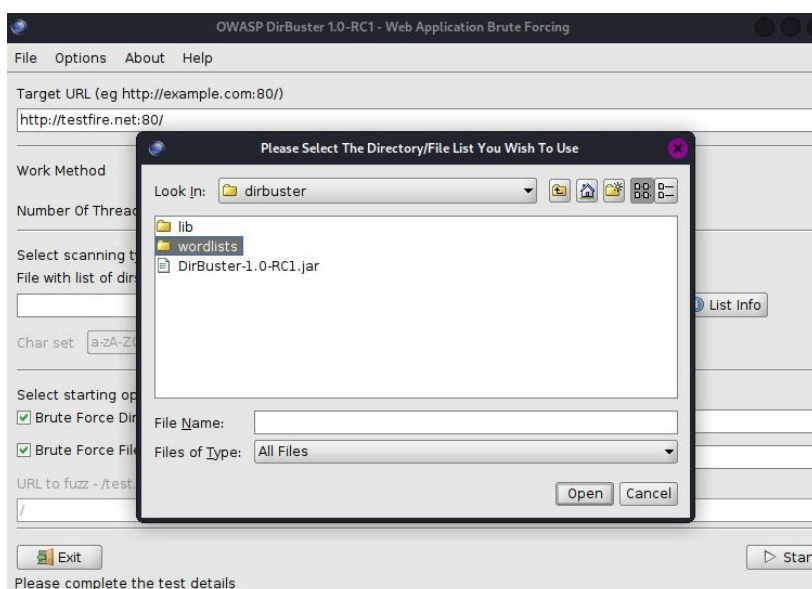
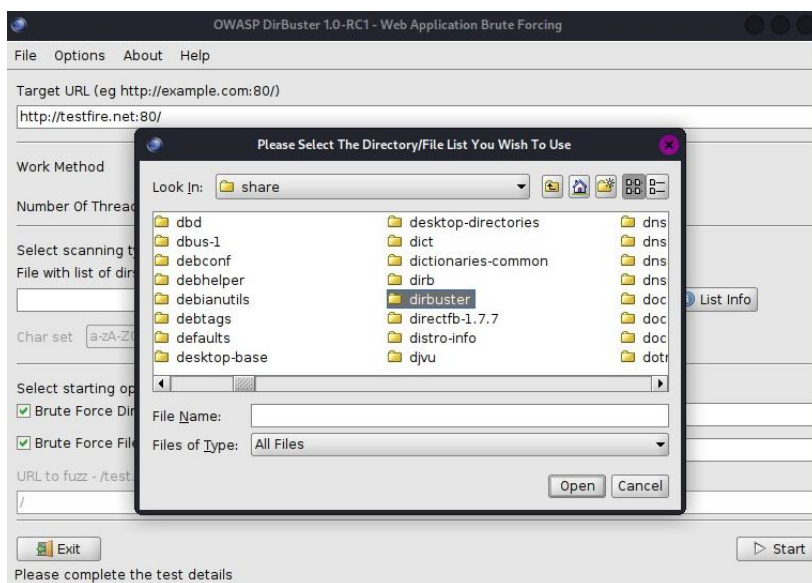
Then we will use the OWASP directory buster to brute force our way through the target website to get the websites directory structure. To use the OWASP directory buster, you can use the following steps. Here our target website will be “www.testfire.net:80/”.

To start dirbuster just type ‘**sudo dirbuster**’ in command line.



Then we will add a list which is already present. Follow the below steps:





OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

Target URL (eg http://example.com:80/)
http://testfire.net:80/

Work Method ☐ Use GET requests only ☒ Auto Switch (HEAD and GET)

Number Of Threads 10 Threads ☐ Go Faster

Select scanning type: ☒ List based brute force ☐ Pure Brute Force

File with list of dirs/files
/usr/share/dirbuster/wordlists/directory-list-1.0.txt

Char set: a-zA-Z0-9%20_ Min length: 1 Max Length: 8

Select starting options: ☒ Standard start point ☐ URL Fuzz

☒ Brute Force Dirs ☒ Be Recursive Dir to start with: /

☒ Brute Force Files ☐ Use Blank Extension File extension: htm

URL to fuzz - /test.html?url={dir}.asp
/

Please complete the test details

Instead of php enter html in file extension.

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://testfire.net:80/

Scan Information Results - List View: Dirs: 8 Files: 10 Results - Tree View Errors: 0

Testing for	Progress	Options
dirs in /	3%	<input type="button" value="Pause"/> <input type="button" value="Stop"/>
files in / with extension .html	4%	<input type="button" value="Pause"/> <input type="button" value="Stop"/>
dirs in /admin/	2%	<input type="button" value="Pause"/> <input type="button" value="Stop"/>
files in /admin/ with extension .html	2%	<input type="button" value="Pause"/> <input type="button" value="Stop"/>
dirs in //	2%	<input type="button" value="Pause"/> <input type="button" value="Stop"/>
files in // with extension .html	2%	<input type="button" value="Pause"/> <input type="button" value="Stop"/>
dirs in /bank/	1%	<input type="button" value="Pause"/> <input type="button" value="Stop"/>

Current speed: 32 requests/sec (Select and right click for more options)

Average speed: (T) 32, (C) 33 requests/sec

Parse Queue Size: 0

Total Requests: 39118/2550547

Current number of running threads: 10

Time To Finish: 21:08:23

Starting dir/file list based brute forcing /ncaa.html

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://testfire.net:80/

Scan Information Results - List View: Dirs: 8 Files: 10 Results - Tree View Errors: 0

Type	Found	Response	Size
Dir	/	200	9560
File	/index.jsp	200	155
File	/login.jsp	200	155
File	/feedback.jsp	200	155
File	/subscribe.jsp	200	155
File	/survey_questions.jsp	200	155
File	/status_check.jsp	200	155
File	/swagger/index.html	200	1716
File	/search.jsp	200	7160
File	/swagger/swagger-ui-standalone-preset.js	200	305722
File	/swagger/swagger-ui-bundle.js	200	935271
Dir	/admin/	302	127
Dir	//	200	155
Dir	/bank/	302	127

Current speed: 39 requests/sec (Select and right click for more options)

Average speed: (T) 32, (C) 34 requests/sec

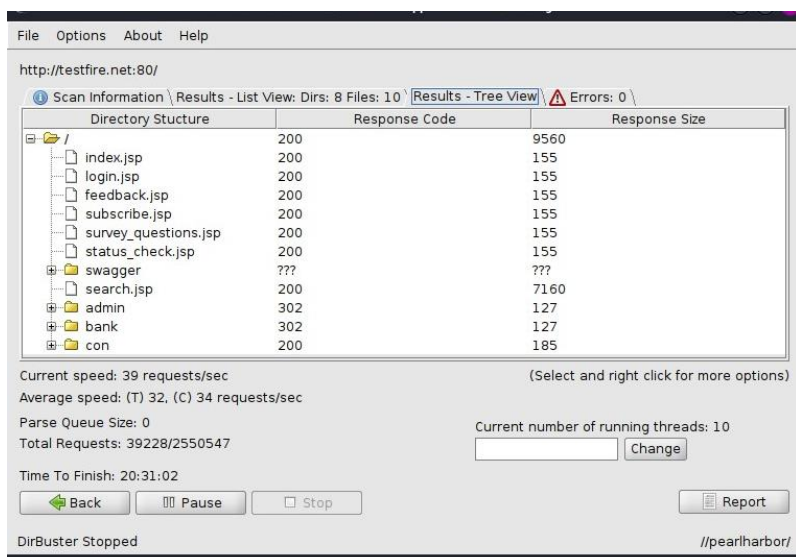
Parse Queue Size: 0

Total Requests: 39228/2550547

Current number of running threads: 10

Time To Finish: 20:31:02

DirBuster Stopped //pearlharbor/



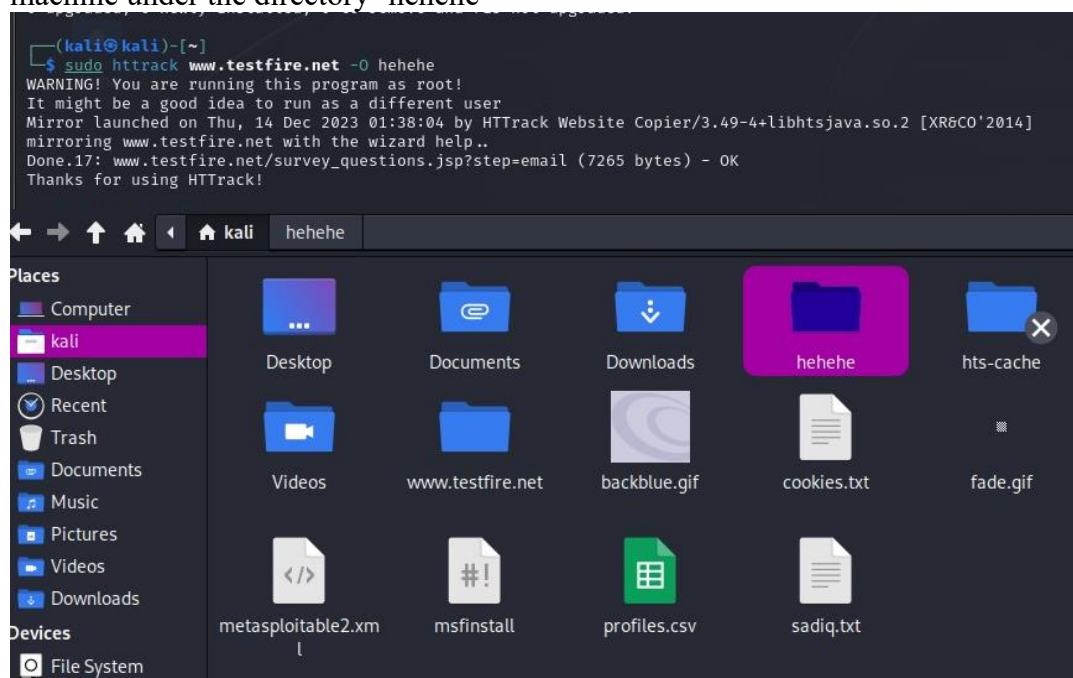
2. Mirroring a website from the command line

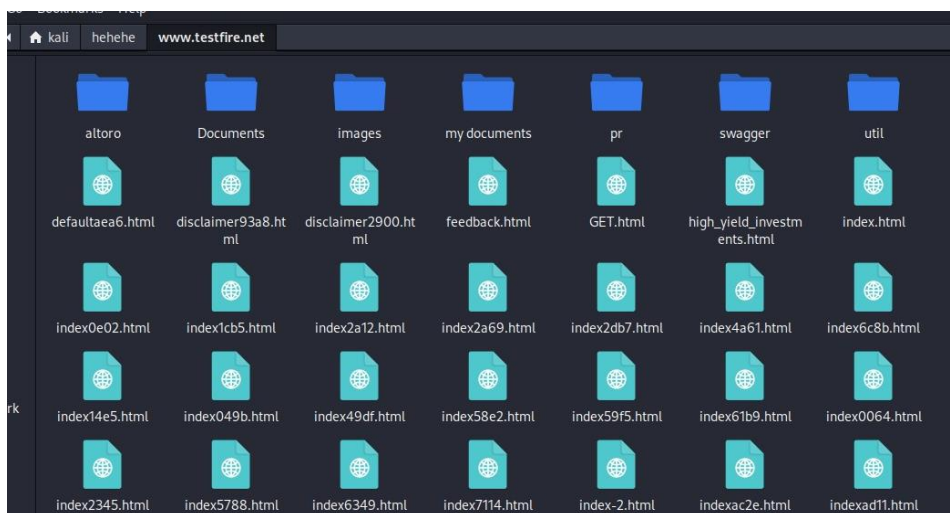
Here we will use HTTRACK, which is an open-source web-crawler that can completely clone a website along with all its directories and its overall file structure.

Since this tool is not a part of Kali Linux, we will have to install it.

```
File Actions Edit View Help
(kali@kali)-[~]
└─$ sudo apt install httrack
[sudo] password for kali:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
httrack is already the newest version (3.49.4-1).
The following packages were automatically installed and are no longer required:
  libmongocrypt0 libncurses5 libtexluaajit2 libtinfo5 lua-lpeg python3-cryptogr
  python3-rx python3-texttable
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 713 not upgraded.
```

Then we will copy our target website www.testfire.net by using this tool and will save it on our machine under the directory 'hehehe'





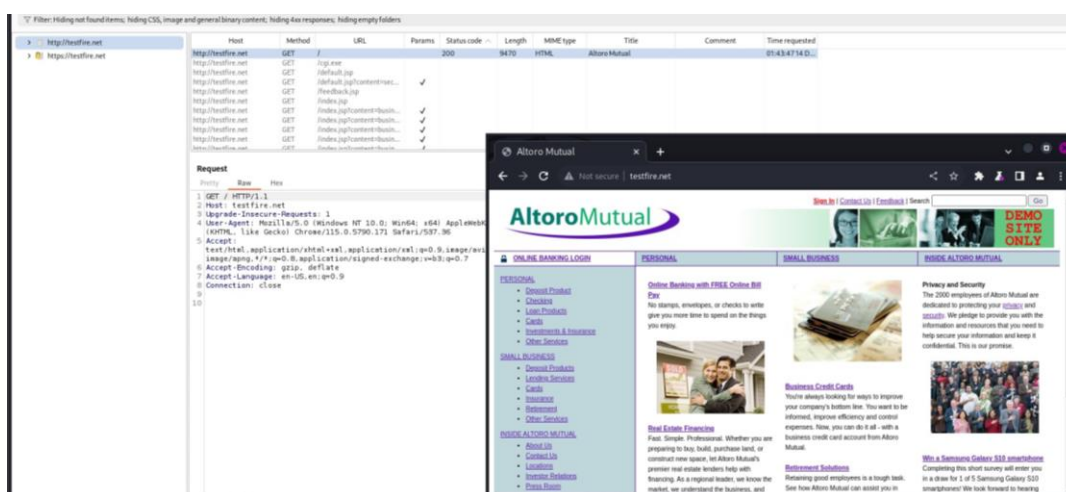
3. Now will use Burp Suite to perform reconnaissance and exploits.

We can access it in the start window of Kali Linux since it comes pre-installed.

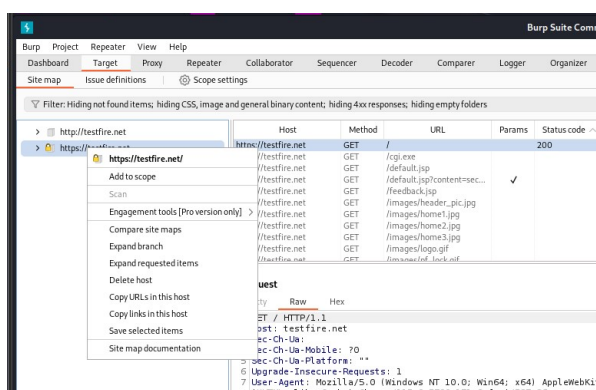
Next, we will create a temporary project.

Then will perform a passive crawl through our target website. Here our target website is “www.testfire.net”. To perform the passive crawl, we have to navigate to the “Target” sub-menu access the in-built browser on the “Sitemap”. We enter our target website into the in-built browser.

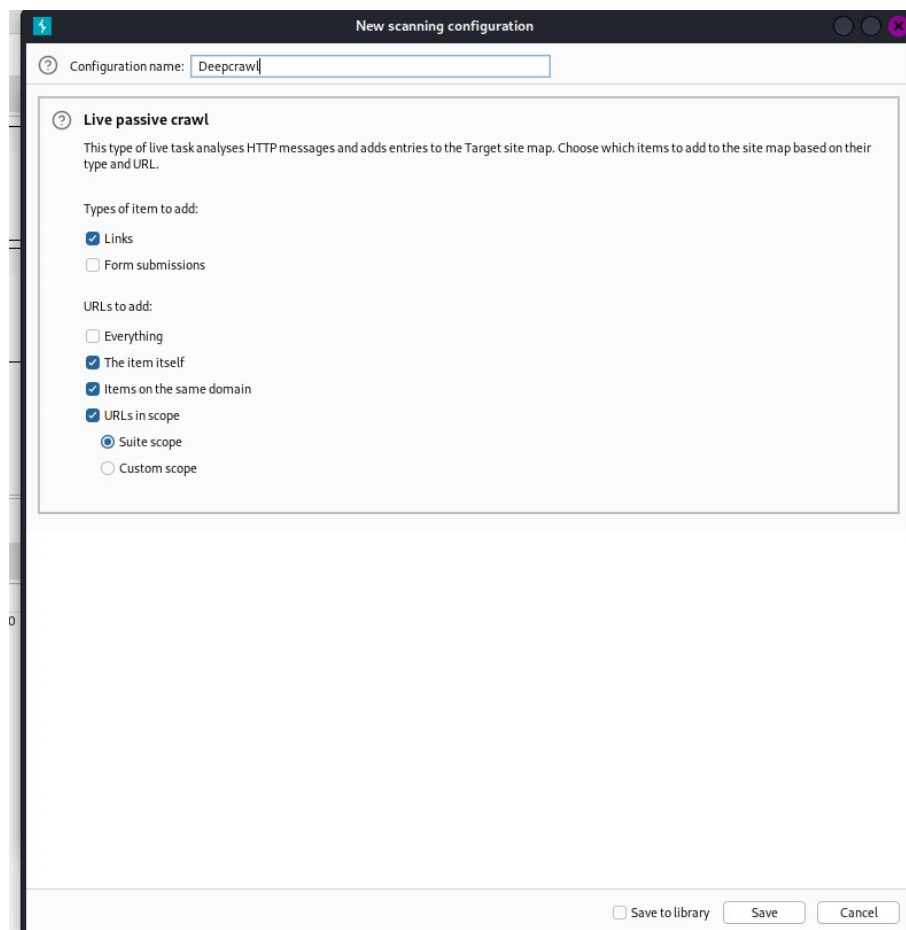
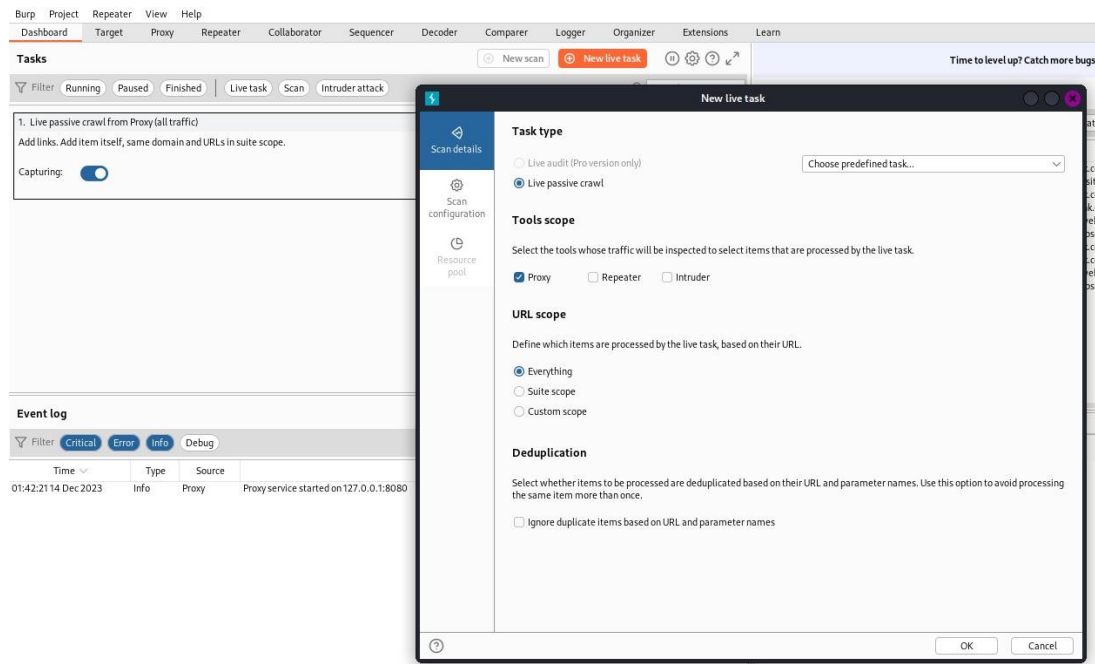
We will start to see the traffic, or the requests being issued on the website in the SiteMap.

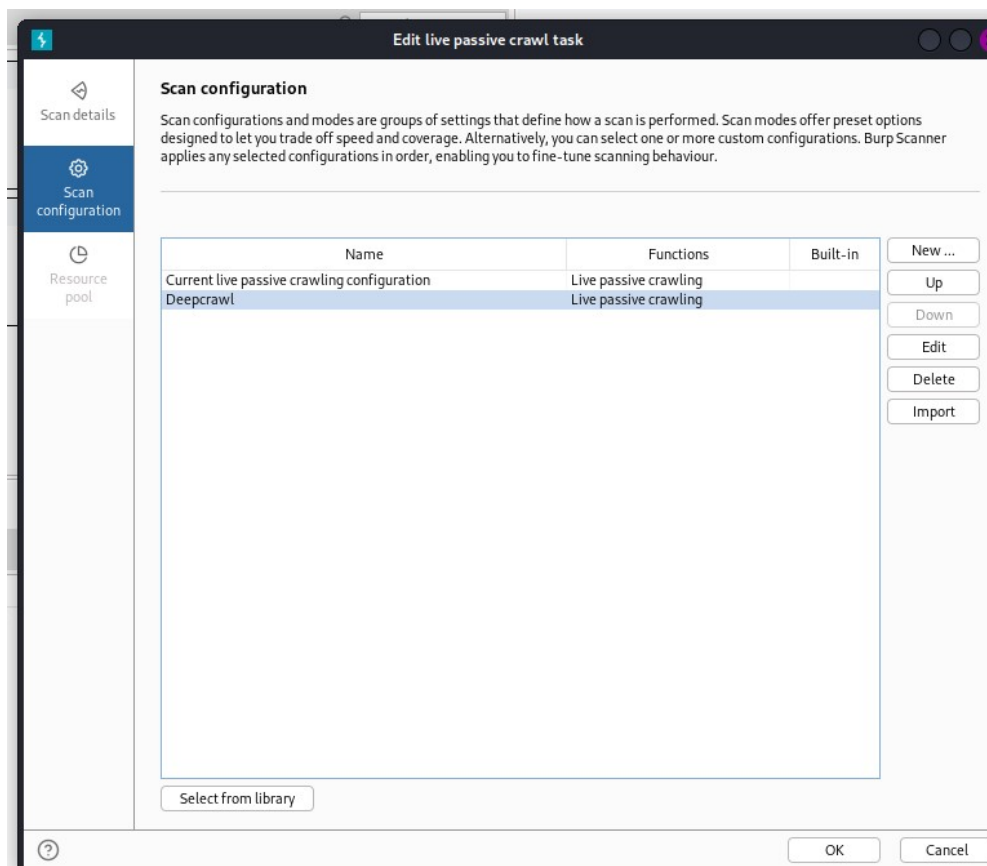


Then we can add the target website to our scope to continue tracking its traffic.

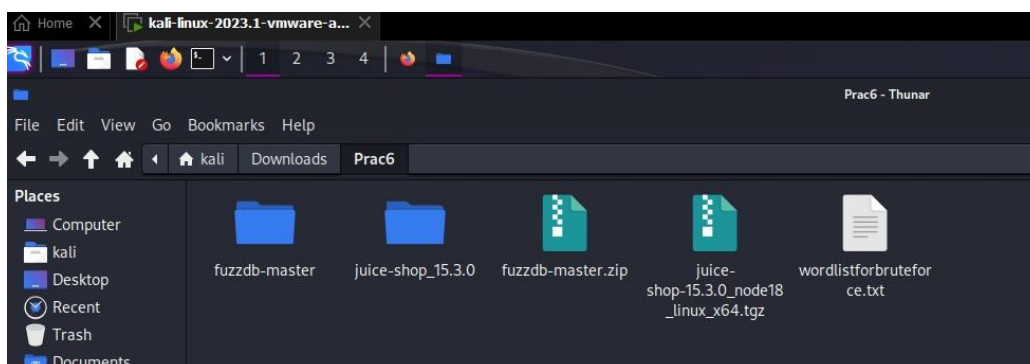


We can also create our own customized passive crawlers by using the following steps.

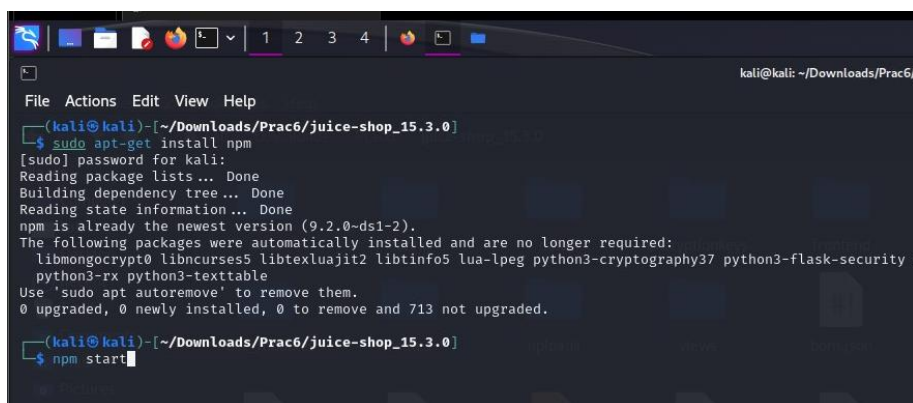




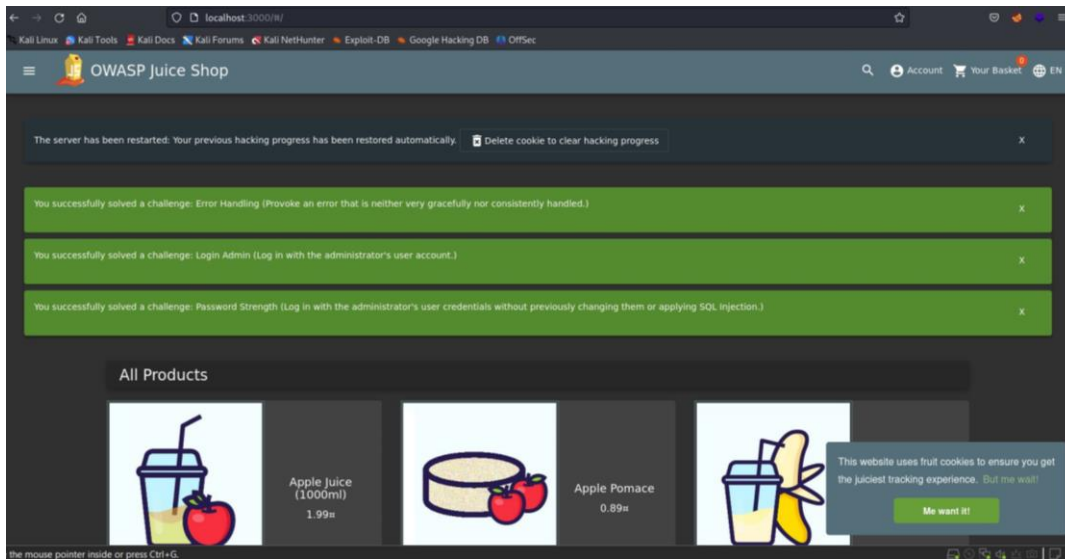
Then we will download fuzzdb master, juice shop and a wordlist and extract it.



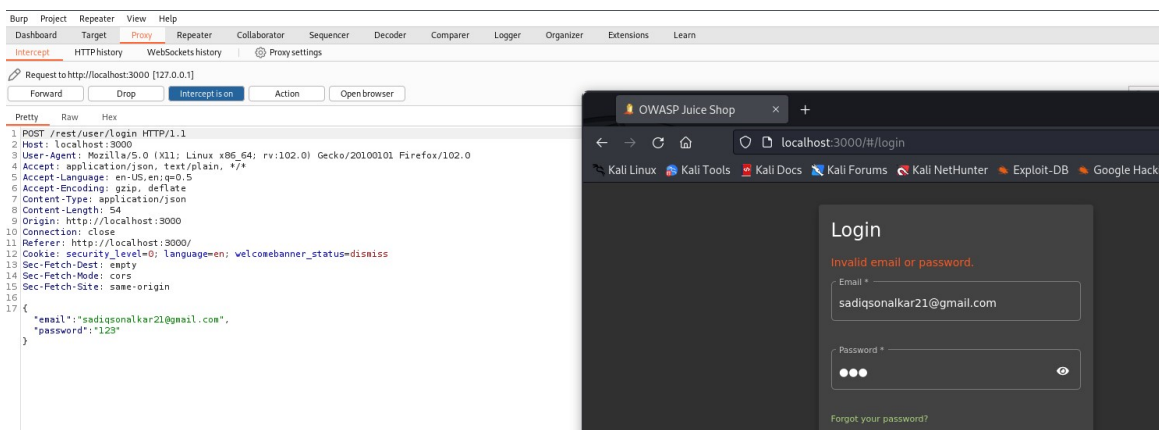
Then in the juice shop folder we will install npm and then start npm.



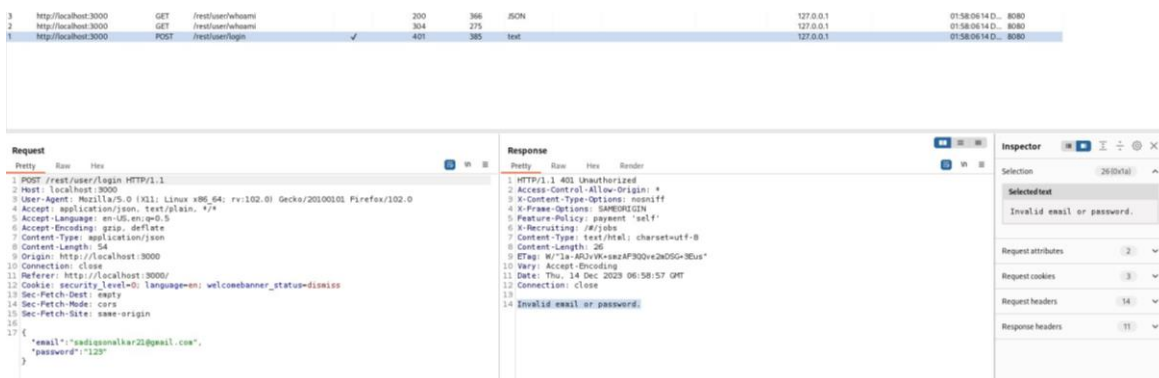
And it will start then go to a browser and enter 'localhost:3000/' and it will show our juice shop webpage.



Turn on the interceptor and try to login using any email id and password.



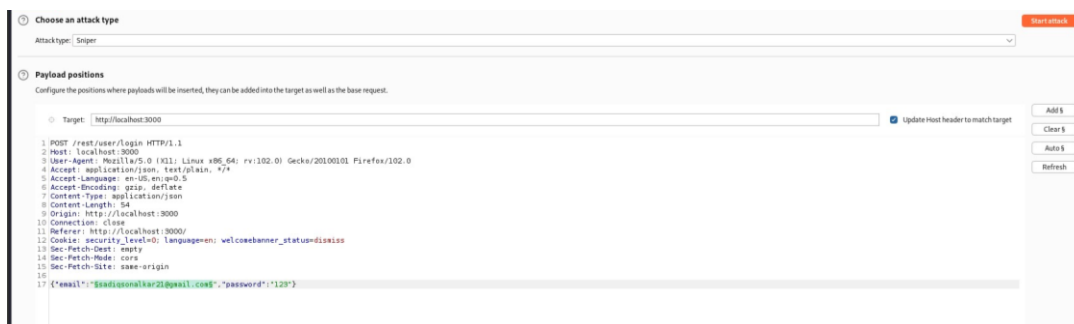
Then go to HTTP history and we can see a post request.



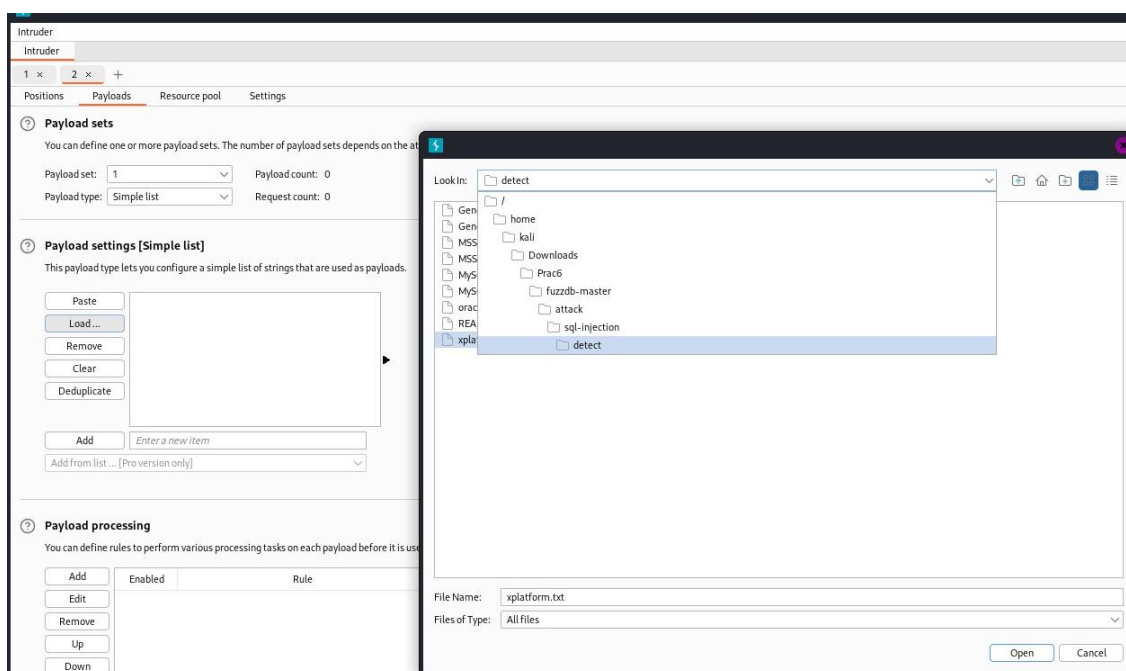
Copy the invalid email or password message.

Right click on the request and then select Send to Intruder.

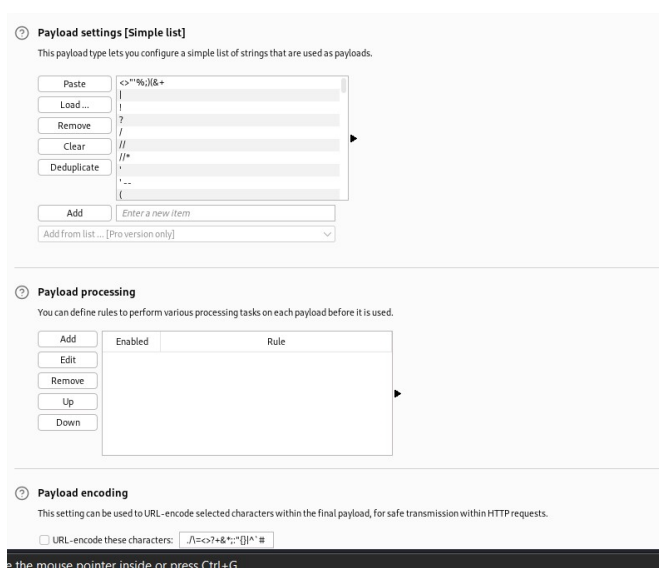
Highlight the parameter that you want to attack, in this case we will target the email to find what email is used for admin login.



Select the payload tab, click on load and then navigate to the wordlist file provided or downloaded to you and select the appropriate option



This should load the strings in the payload. Also uncheck the URL encode option below



Click the mouse pointer inside or press Ctrl+G.

Now select the settings tab and the clear the Grep-Match options and add the text that you had copied earlier. This will help us give a flag if any of our brute force doesn't work it will show the message.

Grep - Match

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

☐ Flag result items with responses matching these expressions:

Paste

Load ...

Remove

Clear


Invalid email or password.

Match type: ☒ Simple string ☐ Regex

☐ Case sensitive match

☒ Exclude HTTP headers

Scroll down a bit and select the In-scope option. This will ignore if there is any caching of pages on the website



Redirections

These settings control how Burp handles redirections when performing attacks.

Follow redirections: ☐ Never ☒ On-site only ☐ Always

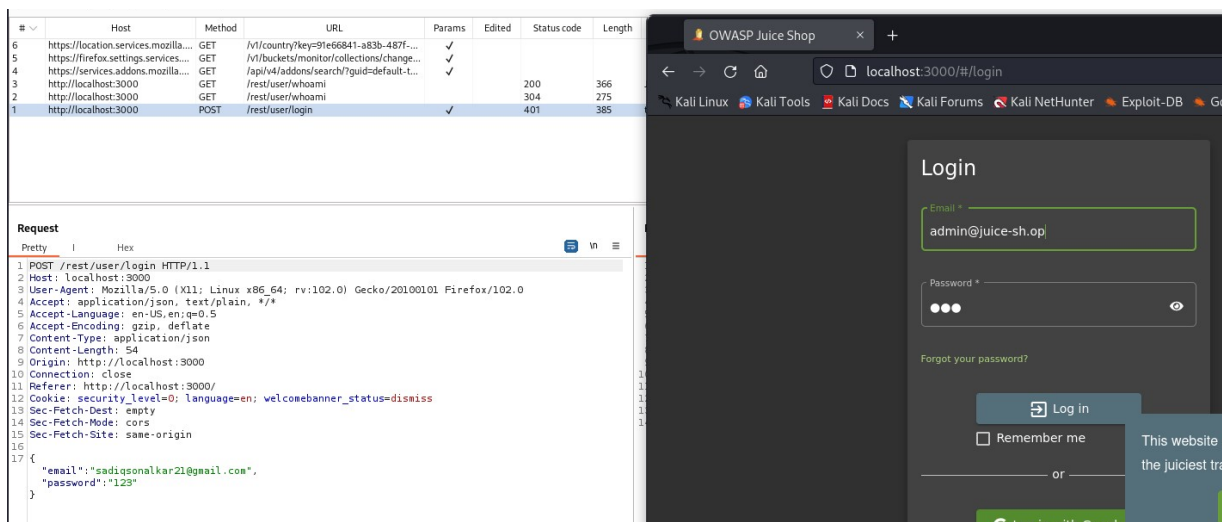
☐ Process cookies in redirections

Now start the attack

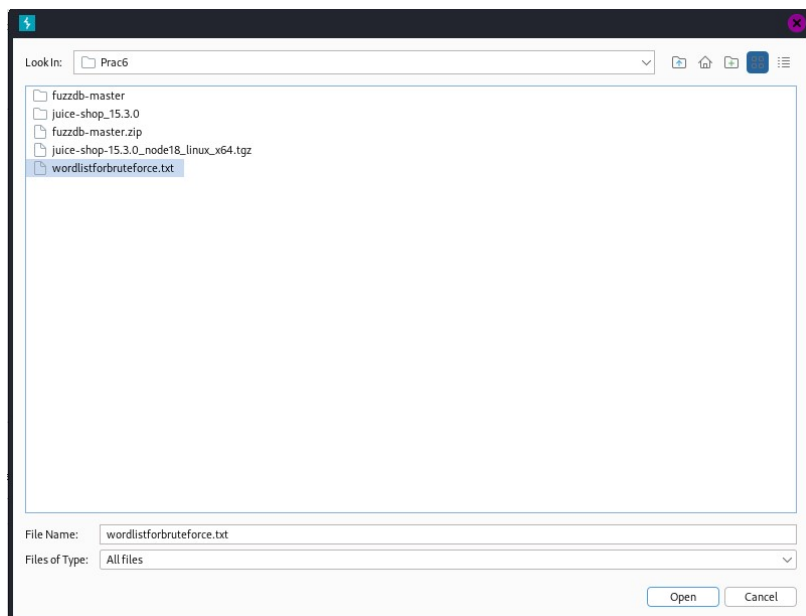
The attack starts. Now search for any request, which gives a 200 status code and token. Within that response you will notice the admin email too. Copy that admin email somewhere.

[illegible]

Now I know the admin email. But I don't know the password. We will need to brute force the login page to get the password. Close the attack session and let's use a wordlist payload to get the password.

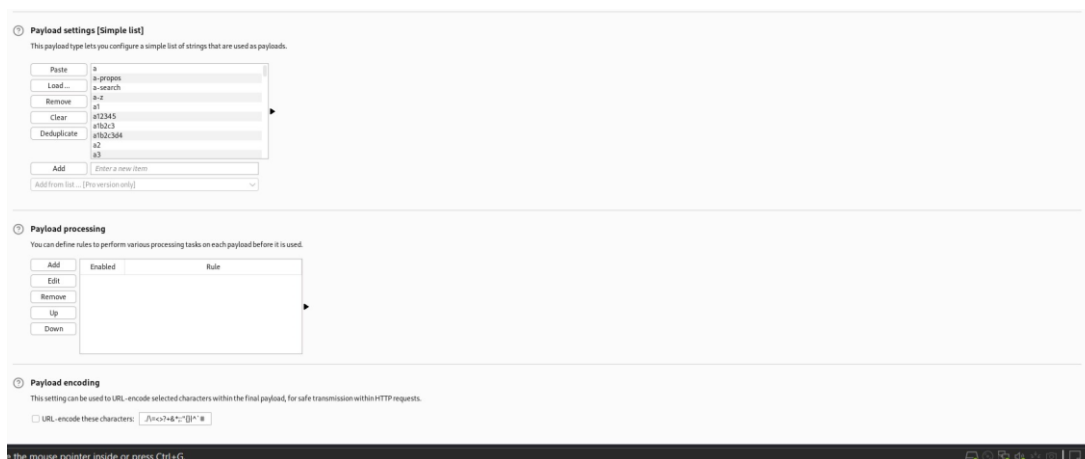


Clear the payload and load a new wordlist payload



Disable the URL encode this character. Then add the invalid email or password we copied before. Also select in-scope only.

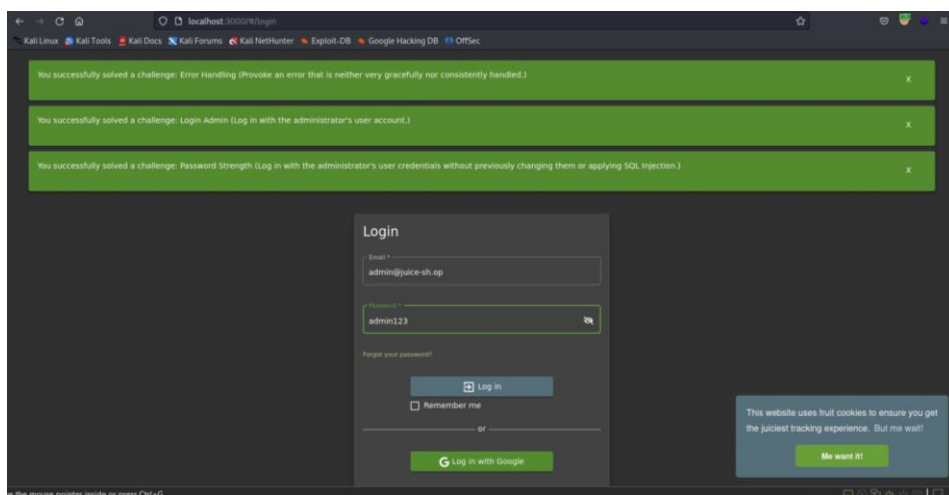
We are ready now to brute force the password field to get the login credentials. Start the attack.



You will notice a 200-response status code on a payload. That payload also does not show Invalid Email or Password error as 1. This means the payload text is the password of the login page.

Attack Save Columns									
Results Positions Payloads Resource pool Settings									
Filter: Showing all items									
Request	Payload	Status code	Error	Redire...	Timeout	Length	Invalid...	Comment	
132	admin	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
133	adminagement	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
134	adminanager	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
135	admbik	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
136	admcp	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
137	admentor	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
138	admi	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
139	admin	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
140	admin-admin	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
141	admin-console	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
142	admin-interface	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
143	admin-login	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
144	admin-old	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
145	admin-panel	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
146	admin00	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
147	admin	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
148	admin12	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
149	admin23	200	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1185			
150	admin2	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
151	admin2009	401	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	413	1		
Request	Response								
Pretty	Raw	Hex							
1 POST /rest/user/login HTTP/1.1									
2 Host: localhost:3000									
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0									
4 Accept: application/json, text/plain, */*									
5 Accept-Language: en-US,en;q=0.5									
6 Accept-Encoding: gzip, deflate									
7 Content-Type: application/json									
8 Content-Length: 51									
9 Origin: http://localhost:3000									
10 Connection: keep-alive									
11 Referer: http://localhost:3000/									
12 Cookie: security_level=0; language=en; welcomebanner_status=dismiss									
13 Sec-Fetch-Dest: empty									
14 Sec-Fetch-Mode: cors									
15 Sec-Fetch-Site: same-origin									
16 {									
17 "email": "admin@juice-sh.op",									
"password": "admin123"									
}									

Now we have got the email and password of the website. Let's try to login the website.



We are now able to access the admin control panel

