

“TechNation Blog” black box penetration testing.

Pen-tester: Monhal Sarbouch.

Author: Monhal Sarbouch,
Monhalsarbouch@gmail.com

Duration: 15/10/2021-25/10/2021

Table of content.

1. Introduction -----	3
1.1 Overview -----	3
1.2 Scope -----	3
1.3 Out of Scope -----	3
1.4 Summary -----	3
2. Conclusion and Improvements -----	4
2.1 Summary of findings -----	4
2.2 Conclusions -----	5
2.3 Improvements -----	5
3. Attack Narrative -----	6

1 introduction

1.1 Overview:

The client asked to perform a complete black-box pen-test on a beta version of the website “TechNation Blog”.

This report documents the vulnerabilities that were found, the methods that were used, screenshots of processes, a conclusion, and suggested fixes that the client should perform in order to secure the web application.

1.2 scope

This security assessment covers the remote penetration testing of 1 accessible server hosted on 10.0.2.10 (internal). the assessment was carried out from a black-box perspective with the only supplied information was the server IP.

Social Engineering was not included in the test scope.

1.3 Out of scope

the client specified that no external resource testing will be permitted, including JS codes that hold URLs that are not part of the domain.

1.4 summary

< this report details the security assessment of “TechNation Blog” website (beta version), the purpose of the assessment was to provide a review of the potential weaknesses of the website.

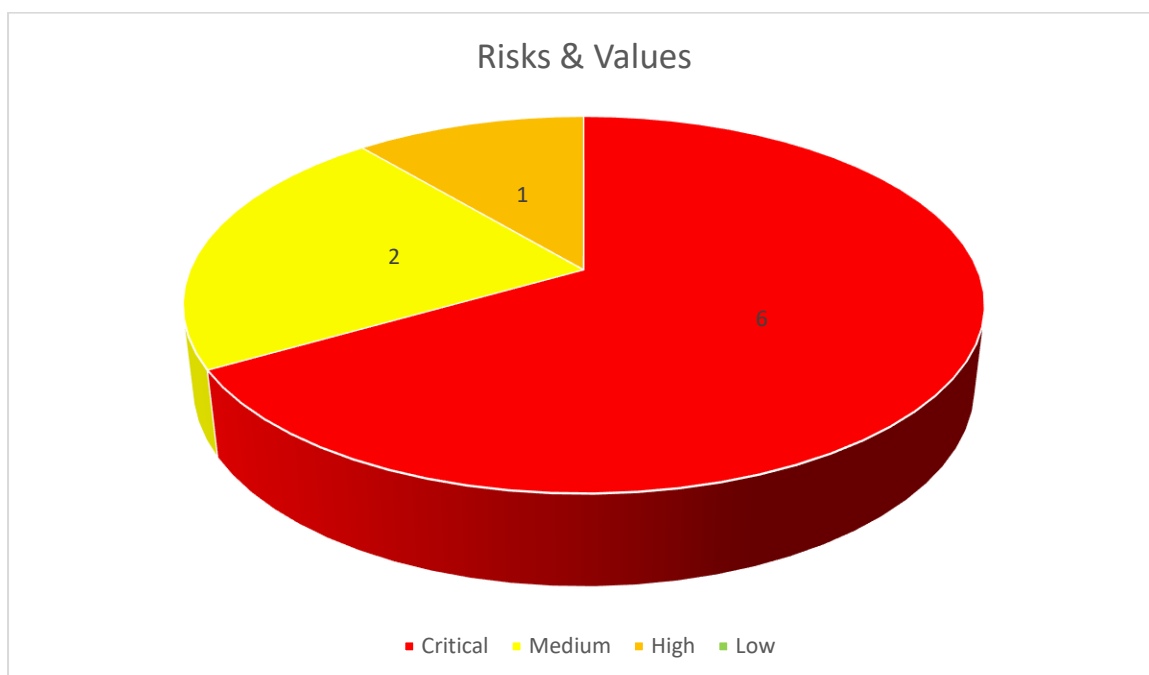
the experience was challenging, not everything went smoothly, a lot of obstracles. SQLi was hard to exploit. >

2. Conclusion And Improvements

2.1 summary of findings

- Visable services and versions (apache 2.4.41, Ubuntu). [medium]
- Plain-text credentials in GET request and in the profile panel. [high]
- Exposed credentials (decoda9013smith21985.txt). [critical]
- Injection (SQLi - password parametes in profile panel). [critical]
- Identification and Authentication Failure (password in profile panel). [critical]
- LFI (Directory Traversal – Site Admin section)[critical]
- RFI. (File Upload – support.php) [critical]
- Injection (XSS, IDOR - Reviews.php, Deals.php). [critical]
- Directory listing. [medium]

Value	Number of Risks
Low	0
Medium	2
High	1
Critical	6



2.2 conclusions

Recording to the supplied assessment, the overall security level of the system is HIGH. The application is highly vulnerable, and can be exploited easily, all of the vulnerability together in one place puts the system in serious danger.

2.3 Improvements

SQLi: this can be solved by applying prepared statements.

XSS: can be fixed with applying htmlentities and/or using WAF.

LFI: this can be fixed by applying a whitelist where external / unauthorized users can access.

RFI: can be fixed by adding `<allow_url_fopen = Off>` and `<allow_url_include = Off>` in php.ini (usually located at etc/php/[php_version]/apache2/php.ini)

Exposed credentials: the decoda9013smith21985.txt file must be removed from the robots.txt or to be moved to a directory that only authorized users can access, and at least to encrypt the passwords with strong Hash function.

Authentication Failure: this can be solved by fixing the 'old' password input in profile panel, and make the input be authenticated by the server's database.

Broken Authentication: can be solved by applying 'captcha' in order to prevent Brute-forcing the login page, also developer must apply strong password.

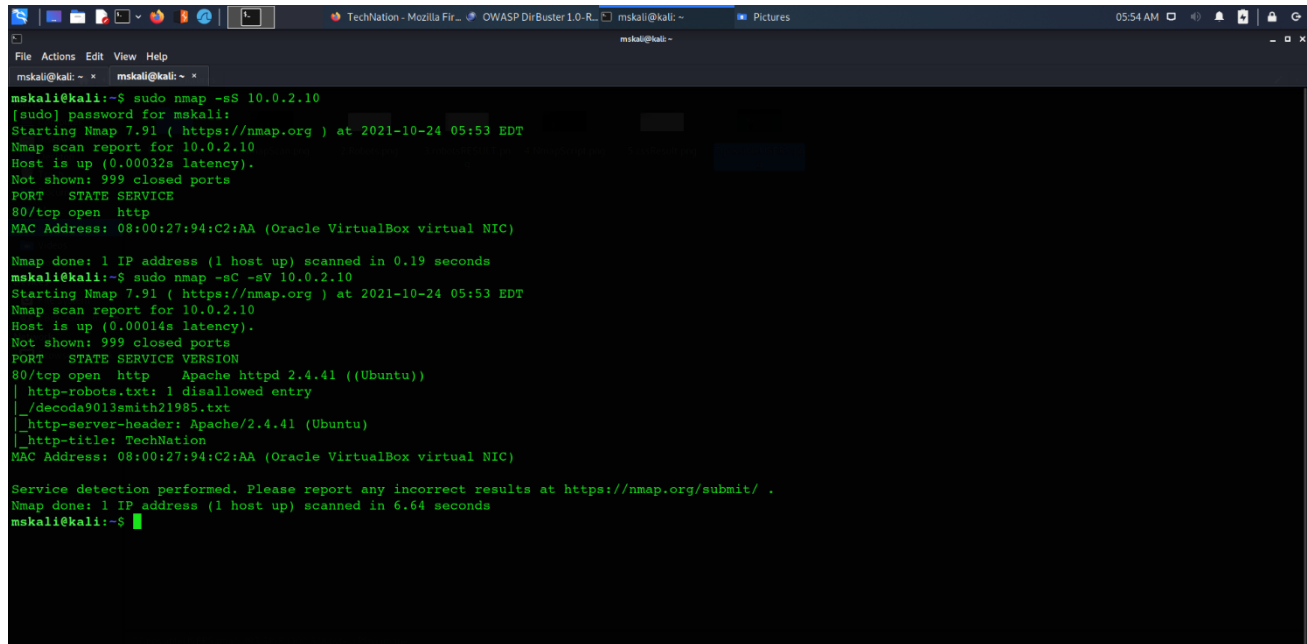
Visible services: can be solved by adding `<ServerTokens Prod>` and `<ServerSignature Off>` in apache configuration file. (usually located at /etc/apache2/apache2.conf)

Directory listing: this option can be disabled in .htaccess file by adding `<Options -Indexes>`

3. Attack Narative:

As I mentioned before, I've been supplied with the website's IP <10.0.2.10> (internal), before accessing the website, I used nmap tool in order to perform information disclosure.

figure 1 (nmap scan)



```

mskali@kali:~$ sudo nmap -sS 10.0.2.10
[sudo] password for mskali:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-24 05:53 EDT
Nmap scan report for 10.0.2.10
Host is up (0.00032s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 08:00:27:94:C2:AA (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds
mskali@kali:~$ sudo nmap -sC -sV 10.0.2.10
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-24 05:53 EDT
Nmap scan report for 10.0.2.10
Host is up (0.00014s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.41 ((Ubuntu))
|_ http-robots.txt: 1 disallowed entry
|_ /decoda9013smith21985.txt
|_ http-server-header: Apache/2.4.41 (Ubuntu)
|_ http-title: TechNation
MAC Address: 08:00:27:94:C2:AA (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.64 seconds
mskali@kali:~$

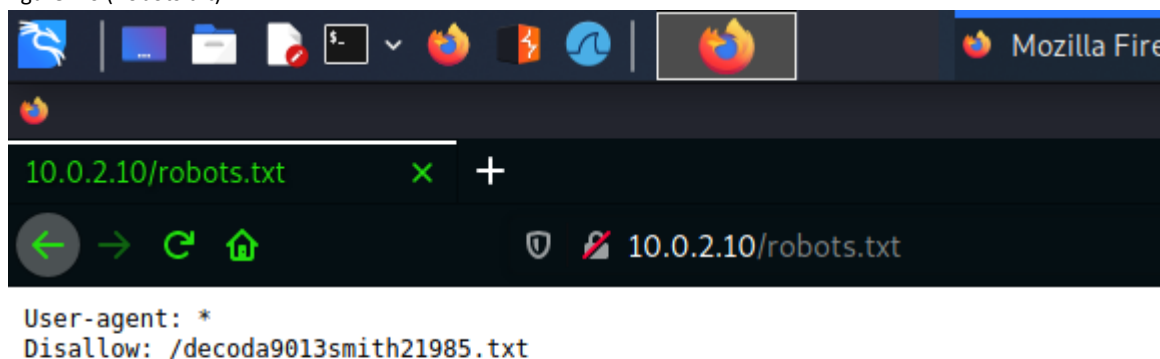
```

the nmap provided me with several important information:

- -Apache Version (2.4.41) and Used OS (Ubuntu)
- decoda9013smith21985.txt file at Robots.txt

Then I navigated to 10.0.2.10/Robots.txt allowing me to locate the file.

figure 2.0 (Robots.txt)



```

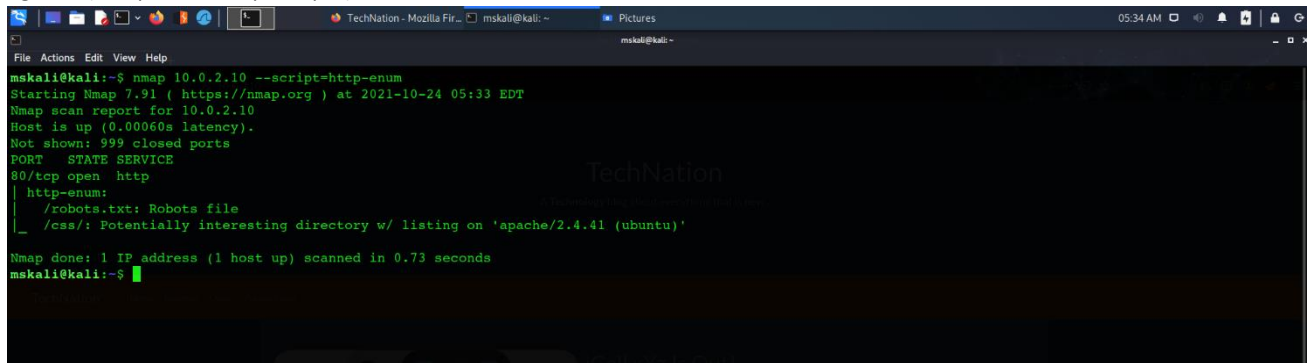
10.0.2.10/robots.txt
User-agent: *
Disallow: /decoda9013smith21985.txt

```


Web Application Penetration Testing Final Project

My next step was using the nmap enumeration script in order to gain more information about the website.

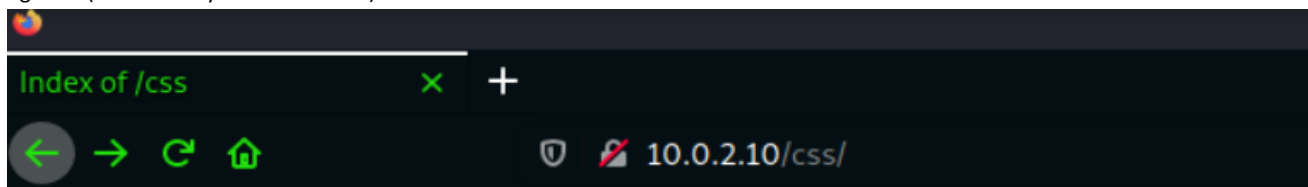
figure 3 (nmap enum-script output)






```
mskali@kali:~$ nmap 10.0.2.10 --script=http-enum
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-24 05:33 EDT
Nmap scan report for 10.0.2.10
Host is up (0.00060s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http
http-enum:
  /robots.txt: Robots file
  /css/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
Nmap done: 1 IP address (1 host up) scanned in 0.73 seconds
mskali@kali:~$
```

This output supplied me with an accessible directory '/css'. Navigated there, nothing interesting was found.

figure 5 (css directory and its content)



Index of /css

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 bootswatch.css	2020-12-27 11:02	208K	
 style.css	2020-12-27 09:39	2.6K	

Apache/2.4.41 (Ubuntu) Server at 10.0.2.10 Port 80

Web Application Penetration Testing Final Project

My next step was to find all possible directories and files that could be accessed. In the step I used DirBuster tool. For directory brute-force. All the found directories and files are attached in the pictures below.

figure 6.0 (site map1 by DirBuster)

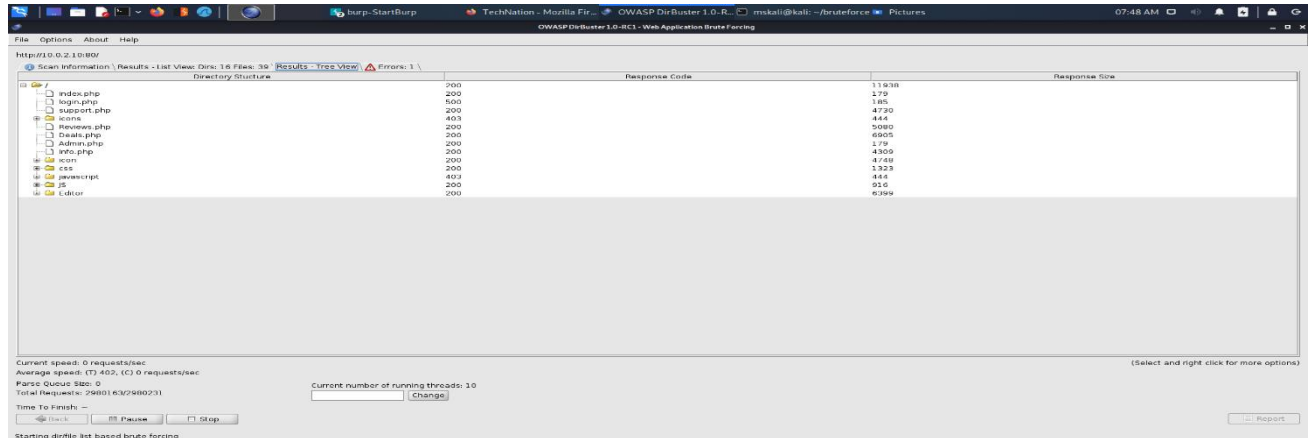


figure 6.1 (site map2 by DirBuster)

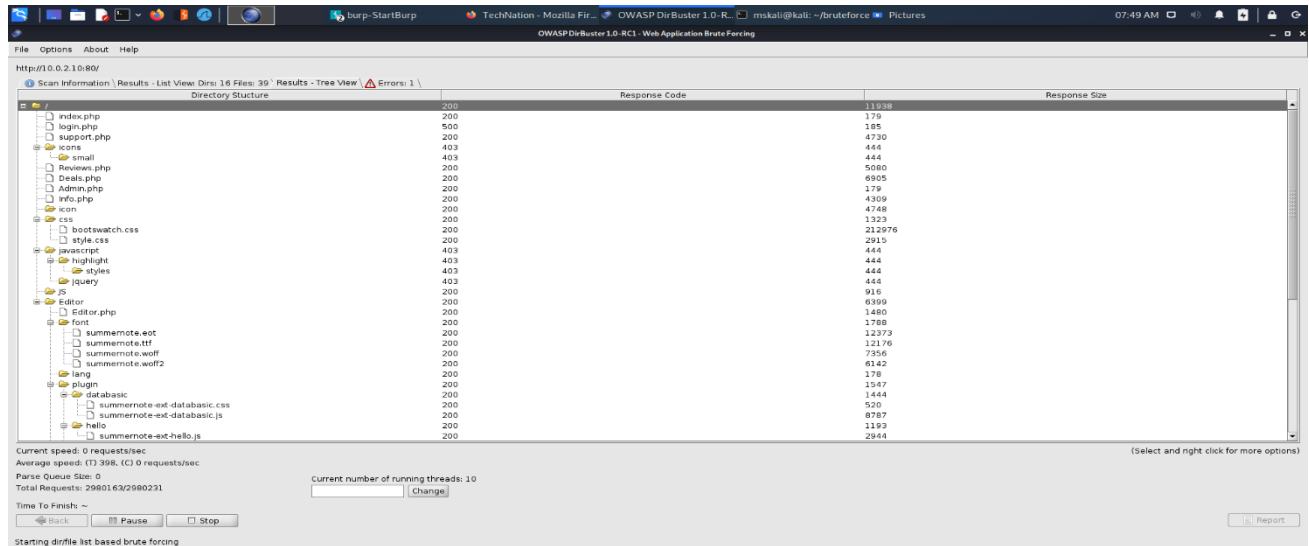
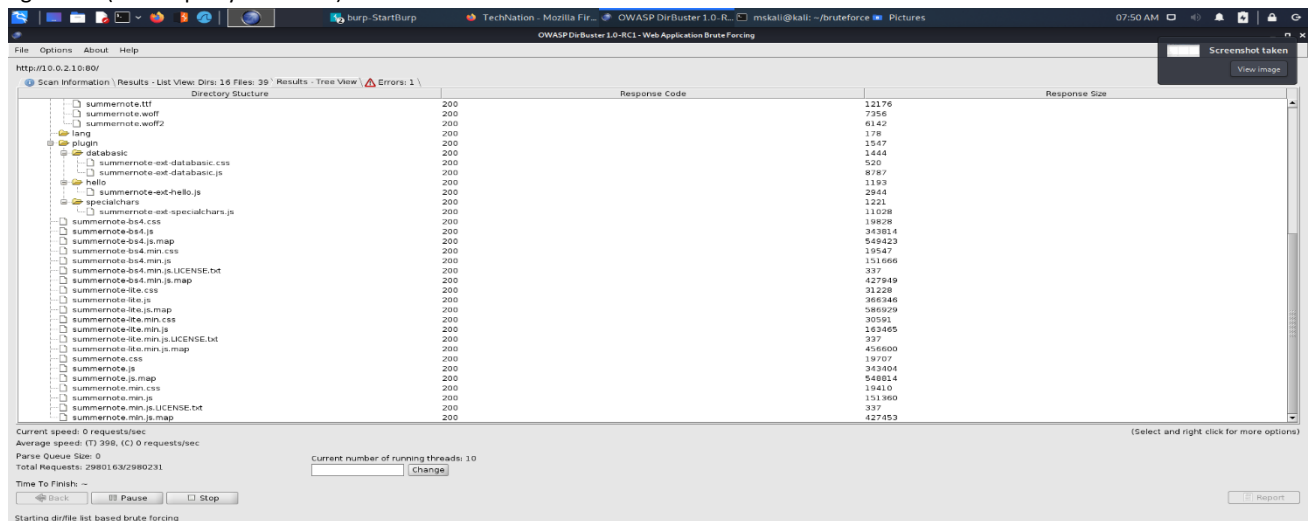


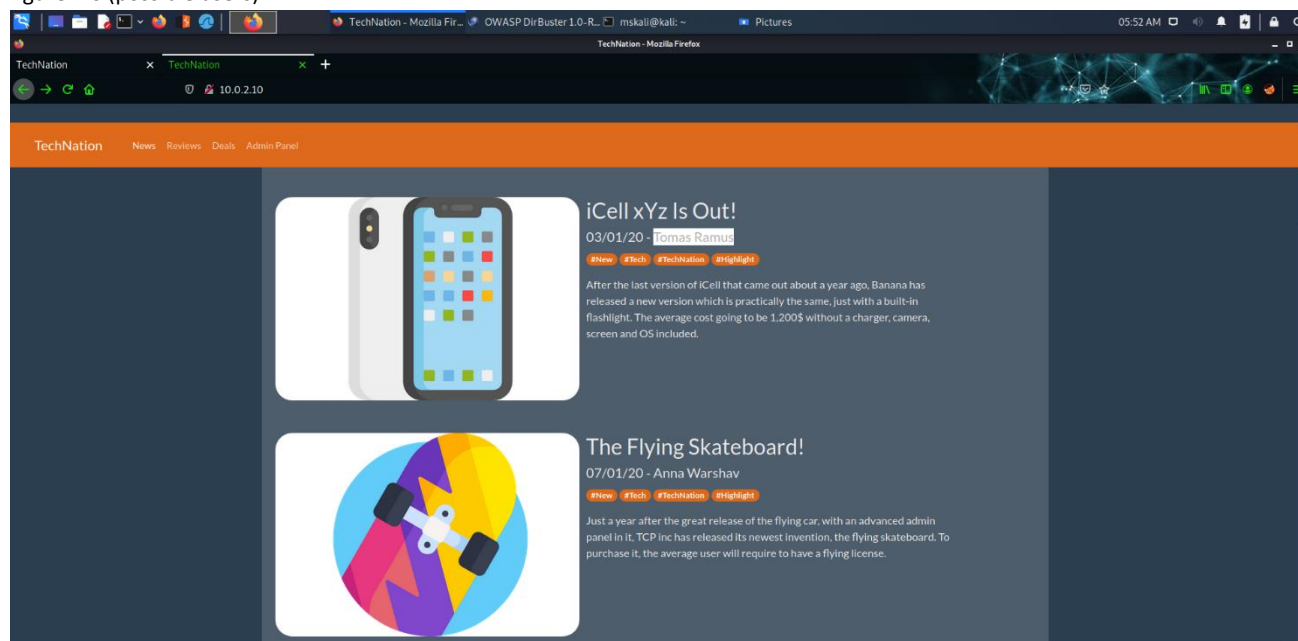
figure 6.2 (site map3 by DirBuster)



Web Application Penetration Testing Final Project

The next step I did was to enumerate users from the website, searched for posts been published by users, found couple possible users.

figure 7.0 (possible users)



The possible users been found are:

- Tomas Ramus
- Anna Warshav
- Arthur Bisclich
- Ronald Copargan

After collecting the users, I've navigated to the 'Admin Panel' in the website and it requested an Email. So I looked in the website for possible domain name, and found About@Technation.com – enough for using as a domain name.

figure7.1 (possible Domain name)

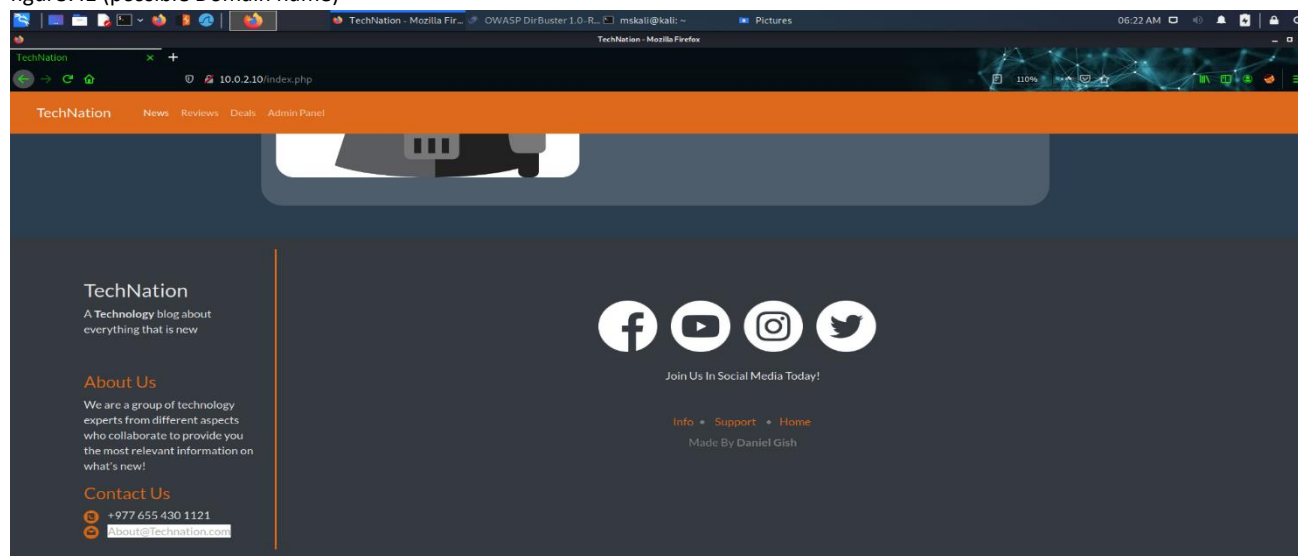
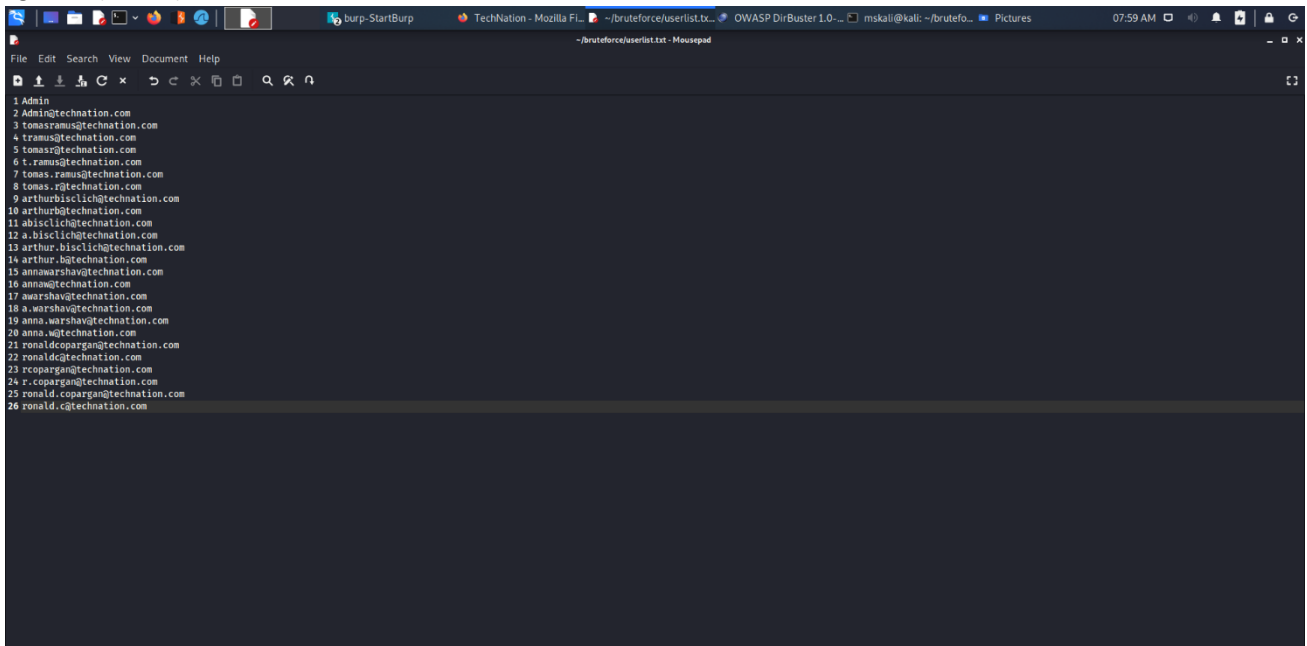
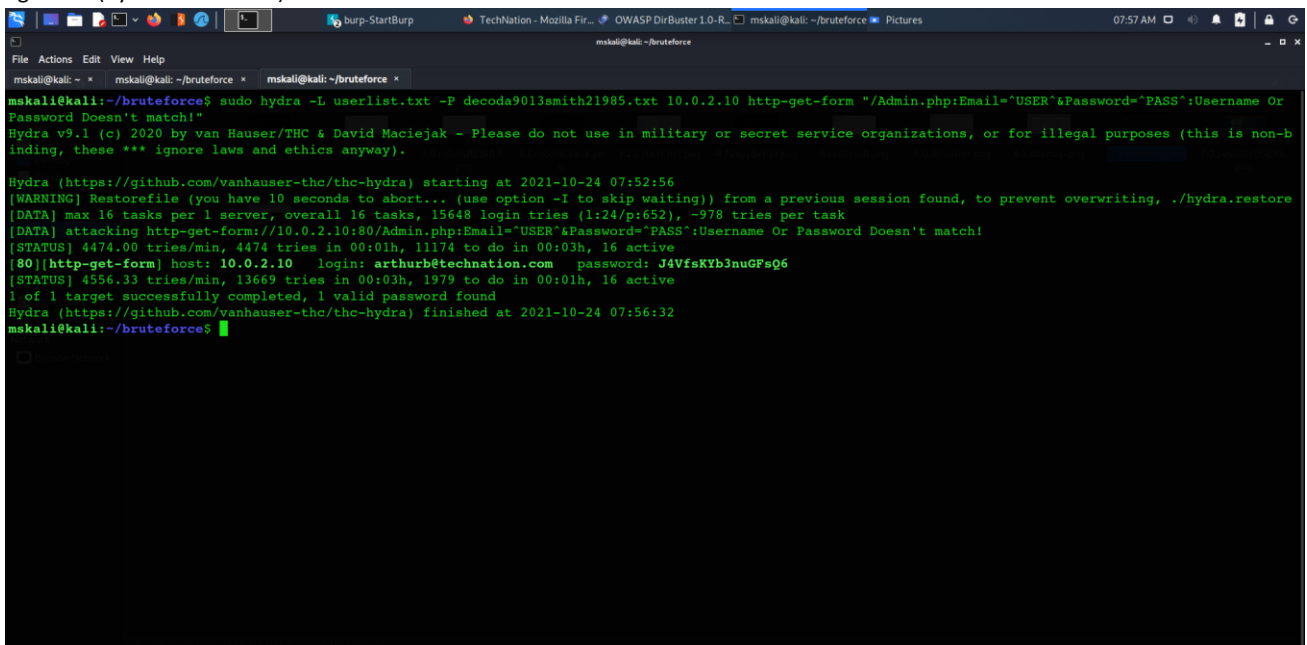


figure7.2 (userlist)



Hydra tool was used to brute-force the login page 'Admin panel'.

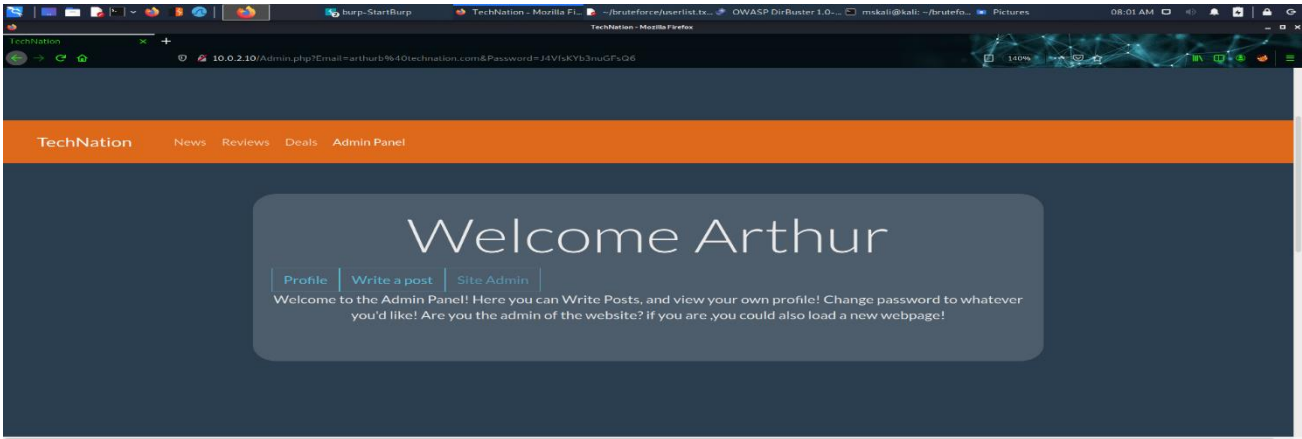
figure7.3 (hydra brute-force)



decoda9013smith21985.txt had a password for the user arthurb@technation.com , after getting the username and the password, I have successfully logged into the admin panel. Noticed that the credentials are in GET method.

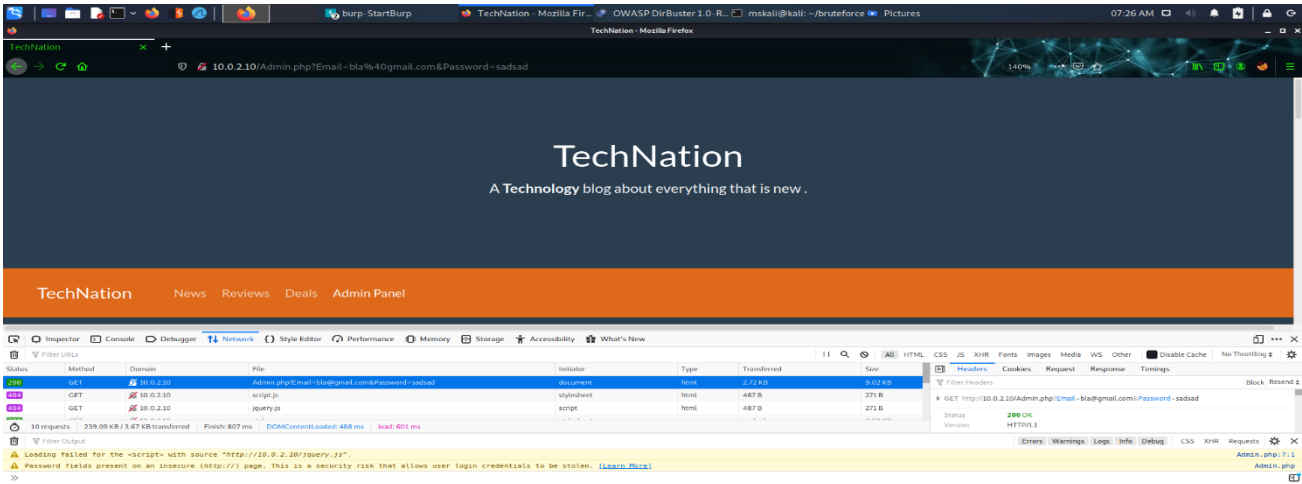
Web Application Penetration Testing Final Project

figure7.4 (successful login & visible credentials in GET)



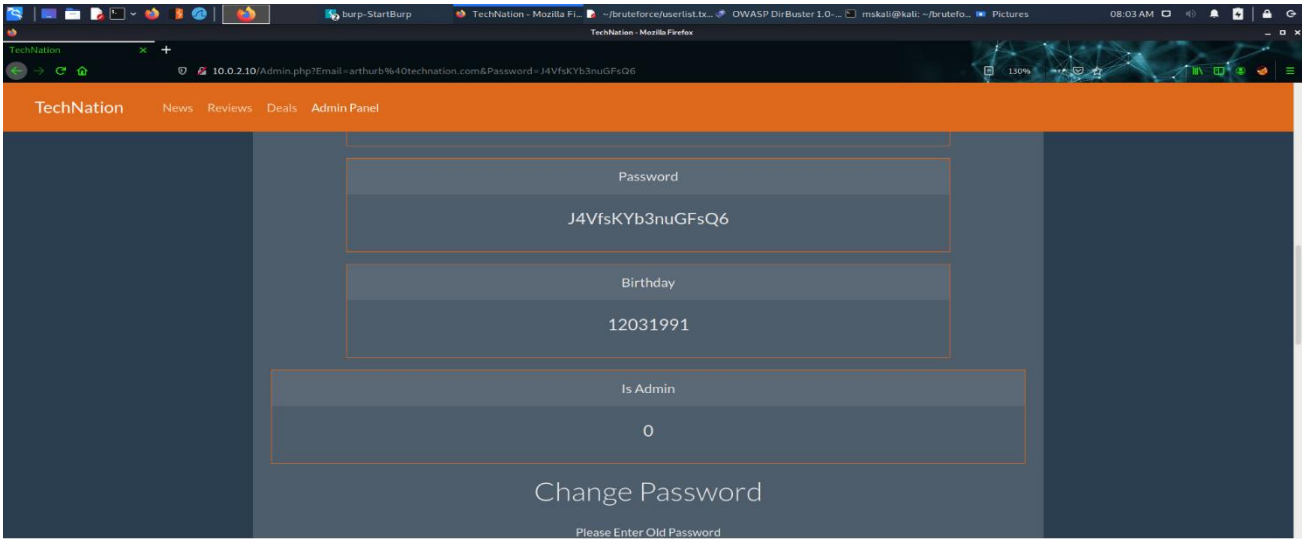
Another proof of credentials in GET:

figure7.5 (credentials in GET method)



After login as arthurb@technation.com, the user was not Admin, another security fail was that the password in the user panel was not hidden.

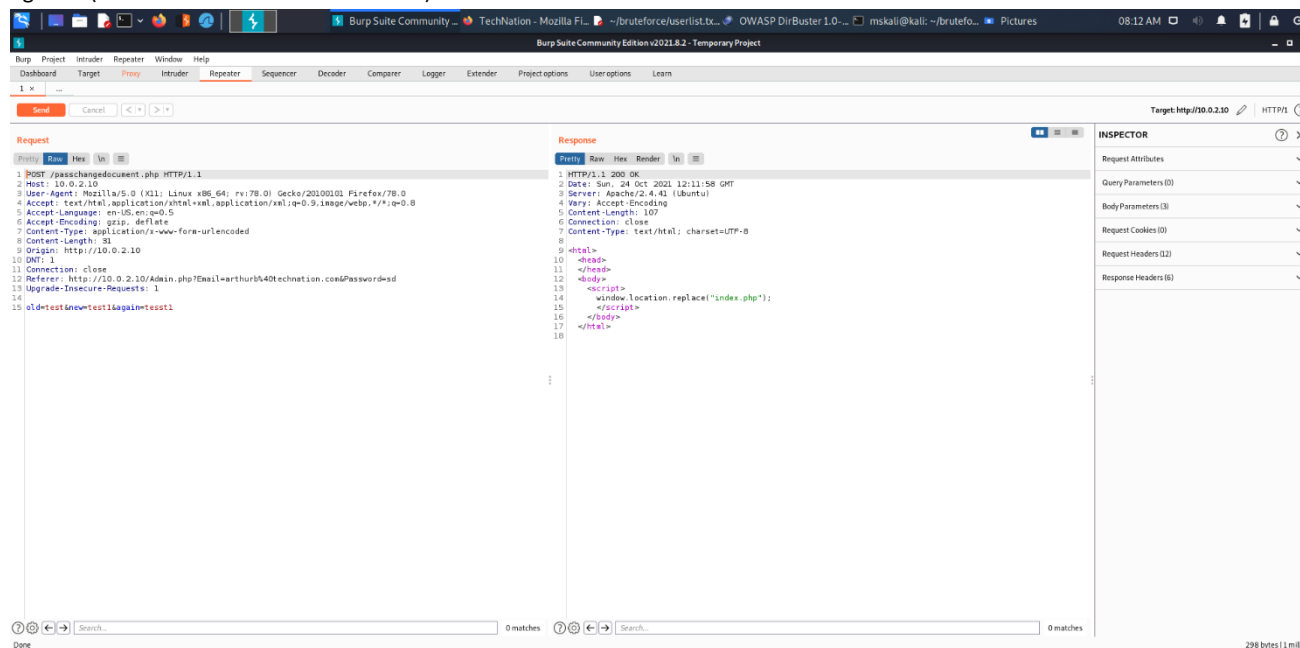
figure7.6 (visible password & not admin)



Web Application Penetration Testing Final Project

One more security failure been found is the 'old' password parameter is not authenticated by the server's database. Which means that no matter what I insert there the password will be changed according to what have been inserted in 'new' and 'again' parametes. In the picture below I inserted wrong password in 'old' parameter and the response was 200.

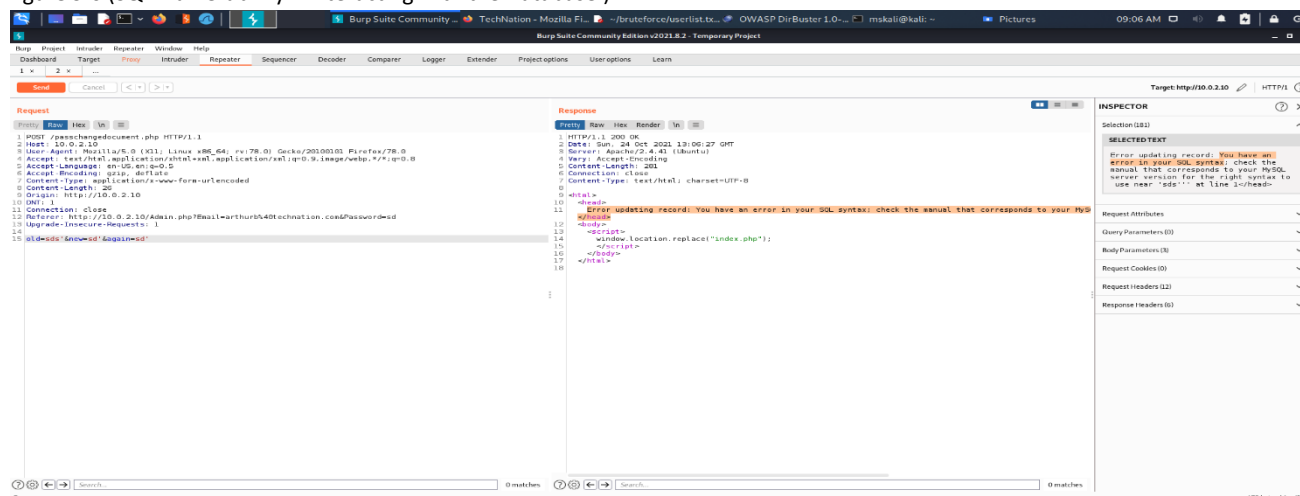
figure 8 (unauthentication with the server)



My next step was checking for SQLi in the parameters on the password change, the parameters were vulnerable to SQLi, so I extracted the request and used sqlmap tool in order to expose the database

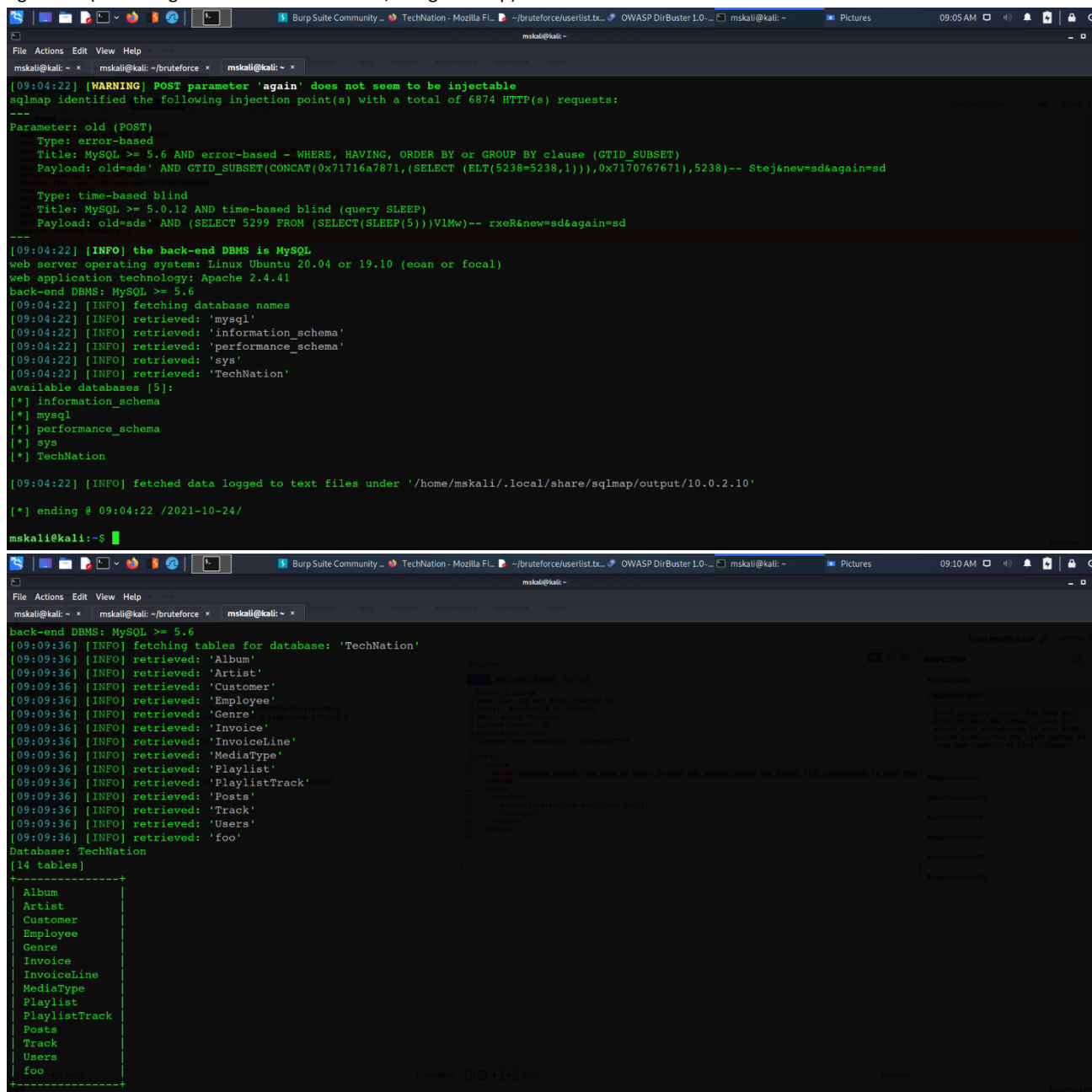
The data base was exposed any I could extract any information from the DB.

figure 9.0 (SQLi vulnerability – interacting with the Database.)



Web Application Penetration Testing Final Project

figure 9.1 (extracting information from the DB, using SQLmap)



After extracting data from 'TechNation' DB, I tried to put a true statement in the 'old' parameter and see what happens, in the picture below I received an error from the DB, so I decoded the true statement to BASE64 using Burp Decoder.

The image displays a Kali Linux desktop environment with three overlapping windows of Burp Suite. The top window, titled 'Burp Suite Community Edition v2021.8.2 - Temporary Project', shows a successful HTTP request and response. The request is a POST to '/passchangedocument.php' with a 'Host' of '10.0.2.10' and a 'User-Agent' of 'Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0'. The response is an HTML page with a status of '200 OK' and a 'Date' of 'Sun, 24 Oct 2021 13:26:30 GMT'. The middle window shows a 401 Unauthorized response. The bottom window shows a successful password update request and response. The request is a POST to '/passchangedocument.php' with a 'Host' of '10.0.2.10' and a 'User-Agent' of 'Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0'. The response is an HTML page with a status of '200 OK' and a 'Date' of 'Sun, 24 Oct 2021 13:26:30 GMT'. The desktop background is a dark blue Linux logo wallpaper. The taskbar at the bottom shows various application icons and the system clock.

That was everything I could exploit in arthurb@techantion.com user, tried to access another users using hydra with 'rockyou.txt' wordlist, and the surprise was that hydra detected all of the passwords withen minutes, all the users password was '1234' and this is according to arthurb@technation.com user password I changed earlier. A critical security missconfiguration leads to compromising all the users registered on the webiste. If changing a password to a user, all the passwords of all the users automatically changes to the specified password.

figure 11 (users are compromised)

```
mskali@kali:~/bruteforce$ sudo hydra -L userlist.txt -P /usr/share/wordlists/rockyou.txt 10.0.2.10 http-get-form "/Admin.php:Email='USER'&Password='PASS':Username Or Password Doesn't match!"
[sudo] password for mskali:
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-theh/the-hydra) starting at 2021-10-27 16:57:07
[DATA] max 16 tasks per 1 server, overall 16 tasks, 71721995 login tries (1:5/p:14344399), ~4482625 tries per task
[DATA] attacking http-get-form://10.0.2.10:80/Admin.php:Email='USER'&Password='PASS':Username Or Password Doesn't match!
[80][http-get-form] host: 10.0.2.10 login: arthurb@technation.com password: 1234
[80][http-get-form] host: 10.0.2.10 login: tomasr@technation.com password: 1234
[80][http-get-form] host: 10.0.2.10 login: annaw@technation.com password: 1234
[STATUS] 43033863.00 tries/min, 43033863 tries in 00:01h, 28688132 to do in 00:01h, 16 active
[80][http-get-form] host: 10.0.2.10 login: ronaldc@technation.com password: 1234
[80][http-get-form] host: 10.0.2.10 login: danielg@technation.com password: 1234
1 of 1 target successfully completed, 5 valid passwords found
Hydra (https://github.com/vanhauser-theh/the-hydra) finished at 2021-10-27 16:58:29
mskali@kali:~/bruteforce$
```

Gained access to all users on the website:

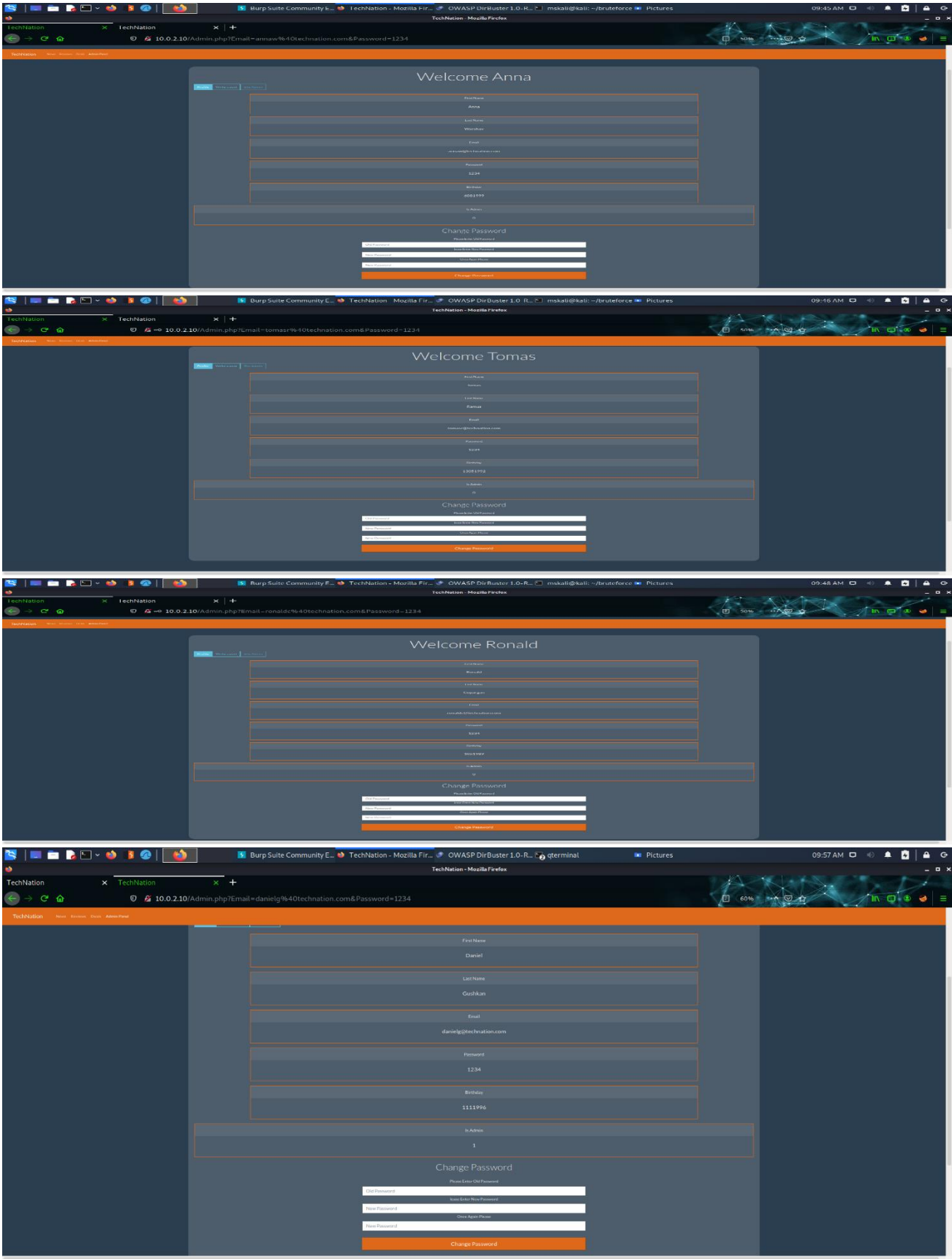
- arthurb@technation.com : 1234 – not Admin
- annaw@technation.com : 1234 – not Admin
- tomasr@technation.com : 1234 – not Admin
- ronaldc@technation.com :1234 – not Admin
- danielg@technation.com :1234 – **Admin**

Then I navigated between the users. Checking for Administrator accounts. Found that danielb@technation.com was the website Admin.

See the picture below.

Web Application Penetration Testing Final Project

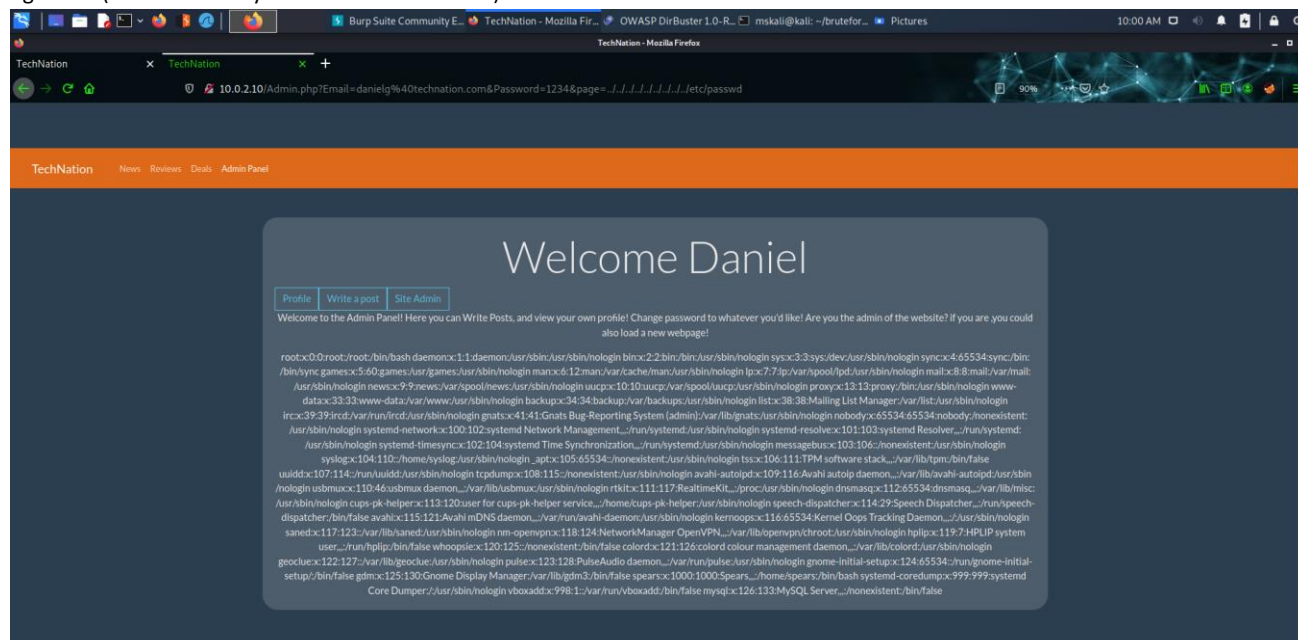
figure 12 (all users are not Admin, except danielb@technation.com)



Web Application Penetration Testing Final Project

After gaining access to danielg@technation.com I start navigating the Admin Panel and found 'Site Admin' section which is vulnerable to LFI, I easily accessed the /etc/passwd file.

figure 13 (LFI vulnerability at 'Site Admin' section.)

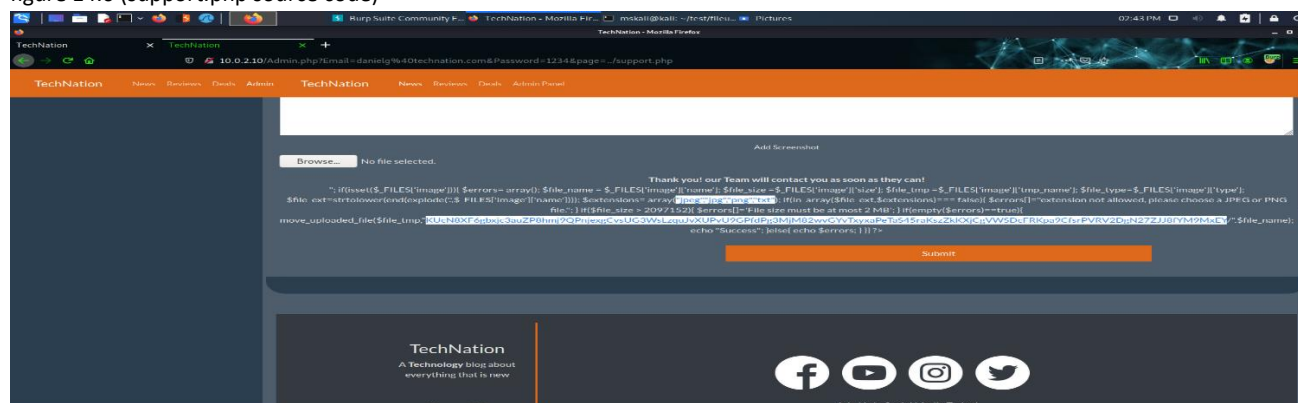


My next step was checking the website pages, found that the 'support.php' page have the function to upload files to the server, so I checked the source code of the page using LFI. Found the extensions accepted and the directory that receives the files.

- Only files with '.jpeg' '.jpg' '.png' '.txt' can be uploaded.
- The directory that accepts the files is :
'KUcN8XF6gbxjc3auZP8hmj9QPnjexgCvsUG3WsLzquJvXUPvU9GPfdPg3MjM82wvGYvTxyxaPeTaS45raKszZkKXjCgVWSDcFRKpa9CfsrPVRV2DgN27ZJJ8fYM9Mx EY'

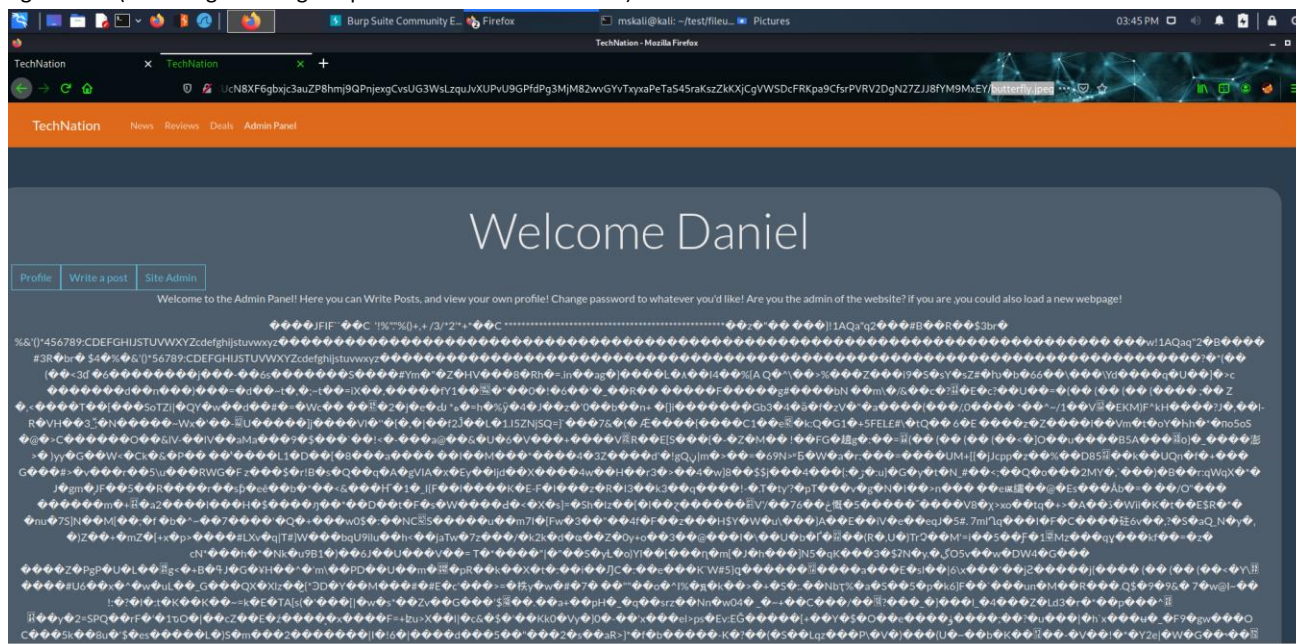
Tired to upload a normal picture (.jpeg) file, and it was successfully uploaded. And I could access it through LFI from the 'Site Admin' section at Admin panel.

figure 14.0 (support.php source code)



Web Application Penetration Testing Final Project

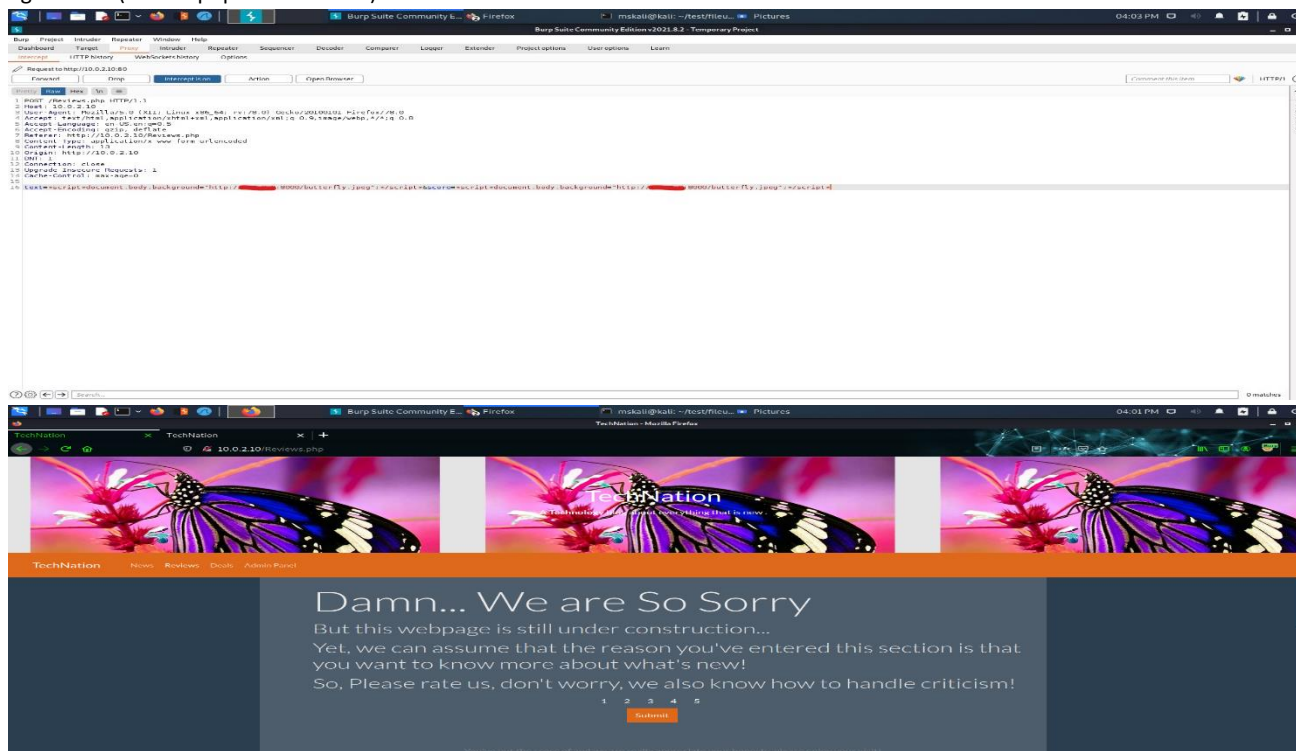
figure 14.1 (accessing the image I uploaded to the server via LFI)



I tried several methods of php file upload bypass (in order to gain RCE), changed the extension of the file, changed the magic bytes and file content ..etc could not find a way to run a .php code.

the next vulnerability exploited was XSS which is reflected in deals.php and reviews.php. using burp suite I was able to change the website's background of the image I uploaded to the server.

figure 15.0 (review.php XSS executed)



I also found that the 'deals.php' is vulnerable to iDOR, the money value could be changed to any number.

figure 15.1 (XSS in Deals.php)

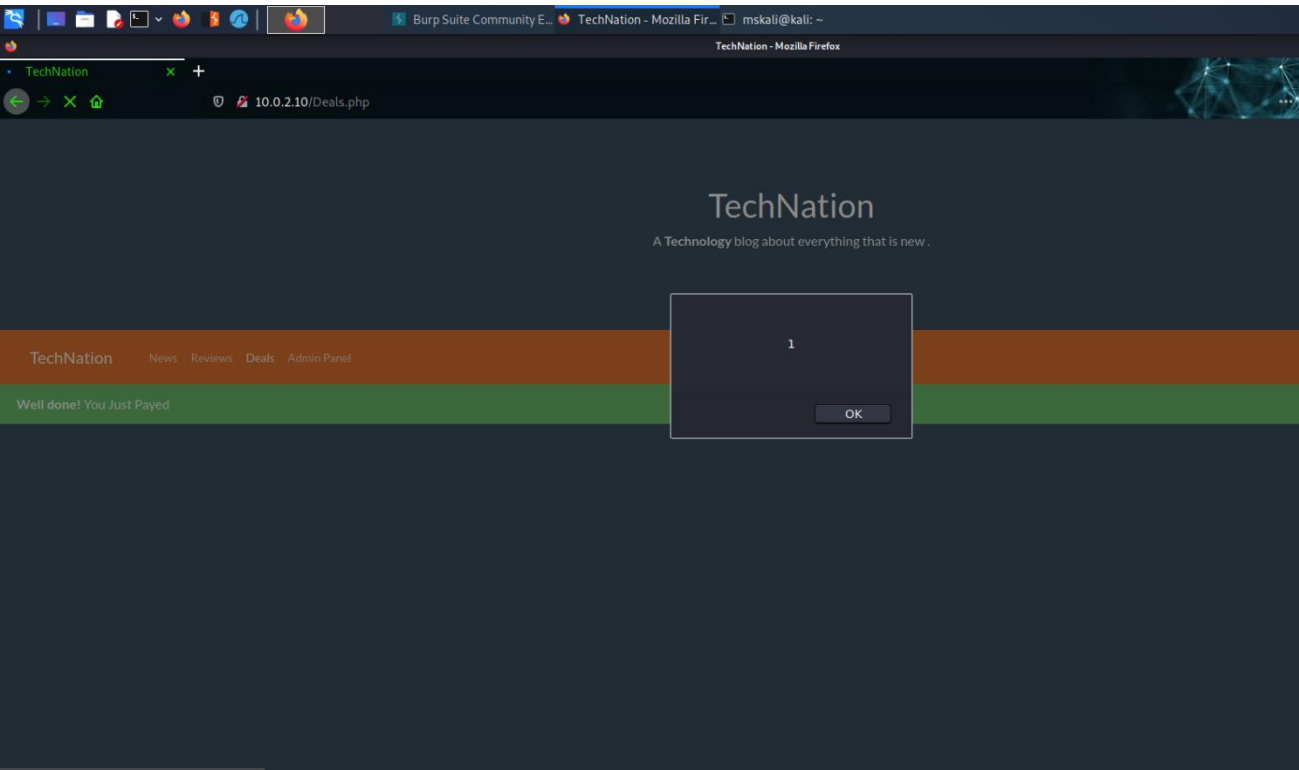


figure 15.2 (iDOR in Deals.php)

