INTERNSHALA
internships that matter

Lifestyle Store

Detailed Developer Report

Submitted by:- Naman Tamboli

Security Status – Extremely Vulnerable

- Hacker can steal all records in Lifestyle Store databases (SQLi)
- Hacker can take control of complete server including View, Add, Edit, Delete files and folders (Shell Upload)
- Hacker can change the entire content of the website(Gaining Admin Control)
- Hacker can inject client side code into applications and trick users by changing how page looks to steal information or spoil the name of Lifestyle Store (XSS)
- Hacker can extract mobile number,address,username and more details of all customers (IDOR)
- Hacker can inject any set of codes and check the server and its content(Command Execution)

Vulnerability Statistics

Critical	Severe	Moderate
10	17	4

Low
8

Vulnerabilities:

No	Severity	Vulnerability	Count
1	Critical	SQL Injection	3
2	Critical	Account takeover via OTP Bypass	1
3	Critical	Access to admin panel	1
4	Critical	Arbitrary File Upload	1
5	Critical	Command Execution	2
6	Critical	Unauthorized Access To Customer Details using IDOR + Rate Limiting Flaw	2
7	Severe	Cross site scripting	4
8	Severe	Cross site request forgery	2
9	Severe	Rate Limiting Flaws	7
10	Severe	Crypto Configuration Flaw	1
11	Severe	Weak Passwords Policy	2
12	Severe	Open Redirection	1

Vulnerabilities:

No	Severity	Vulnerability	Count
13	Moderate	PII Leakage	1
14	Moderate	Components with Known Vulnerabilities	2
15	Moderate	Forced Browsing Flaw	1
	Low	Forced Browsing Flaw	4
16	Low	Directory Listing	1
17	Low	Default error display	2
18	Low	Client Side Filter Bypass	1

1. SQL Injection

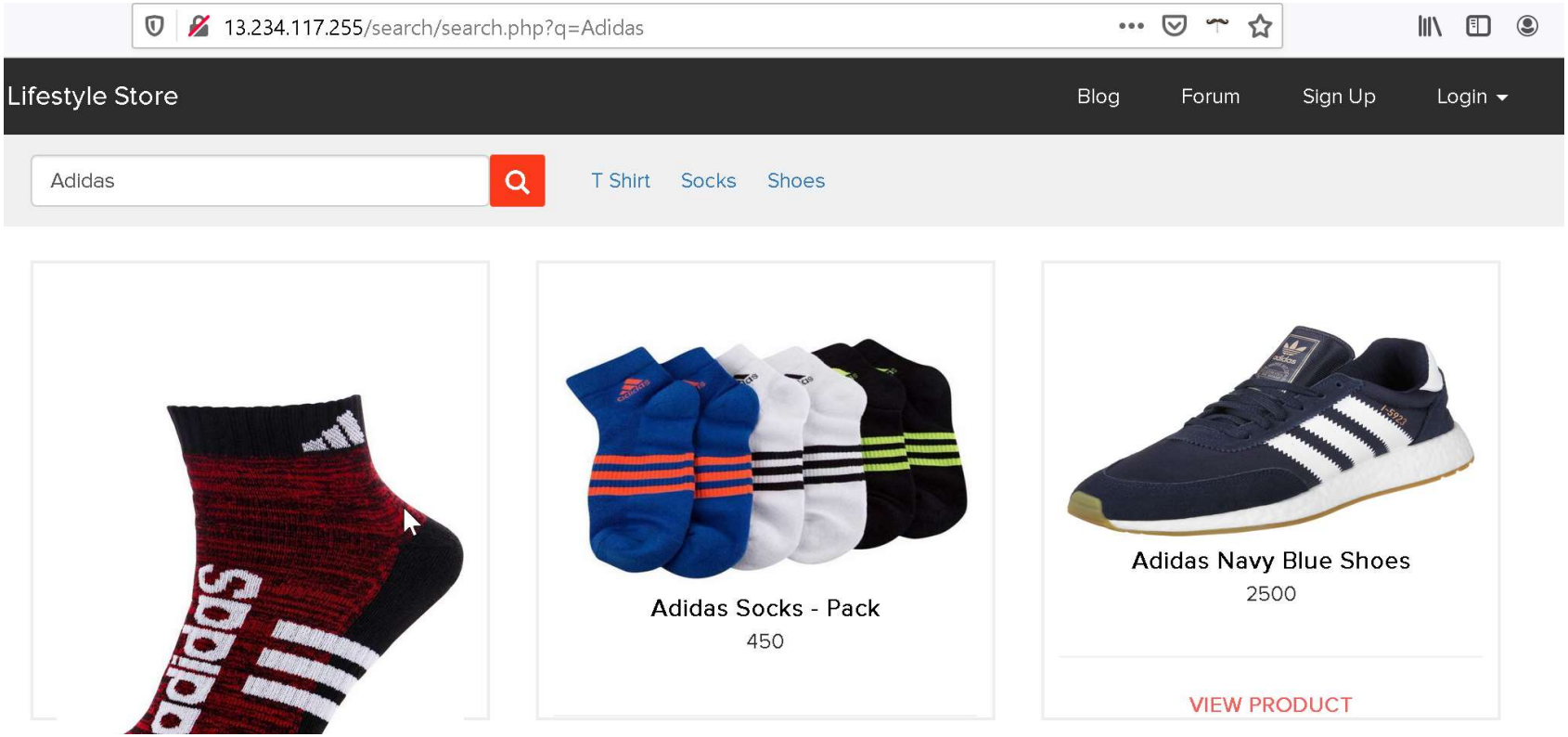
SQL Injection (Critical)	Below mentioned URL is vulnerable to SQL injection attack **Affected URL :** • http://13.234.117.255/search/search.php?q=Adidas **Affected Parameters :** • q (GET parameter) **Payload:** • q=Adidas’

1. SQL Injection

SQL Injection (Critical)	Here are other similar SQLi in the application **Affected URL :** • http://13.234.117.255/products.php (page ,POST parameter) • http://13.234.117.255/products.php?cat=1(cat,GET parameter)

Observation

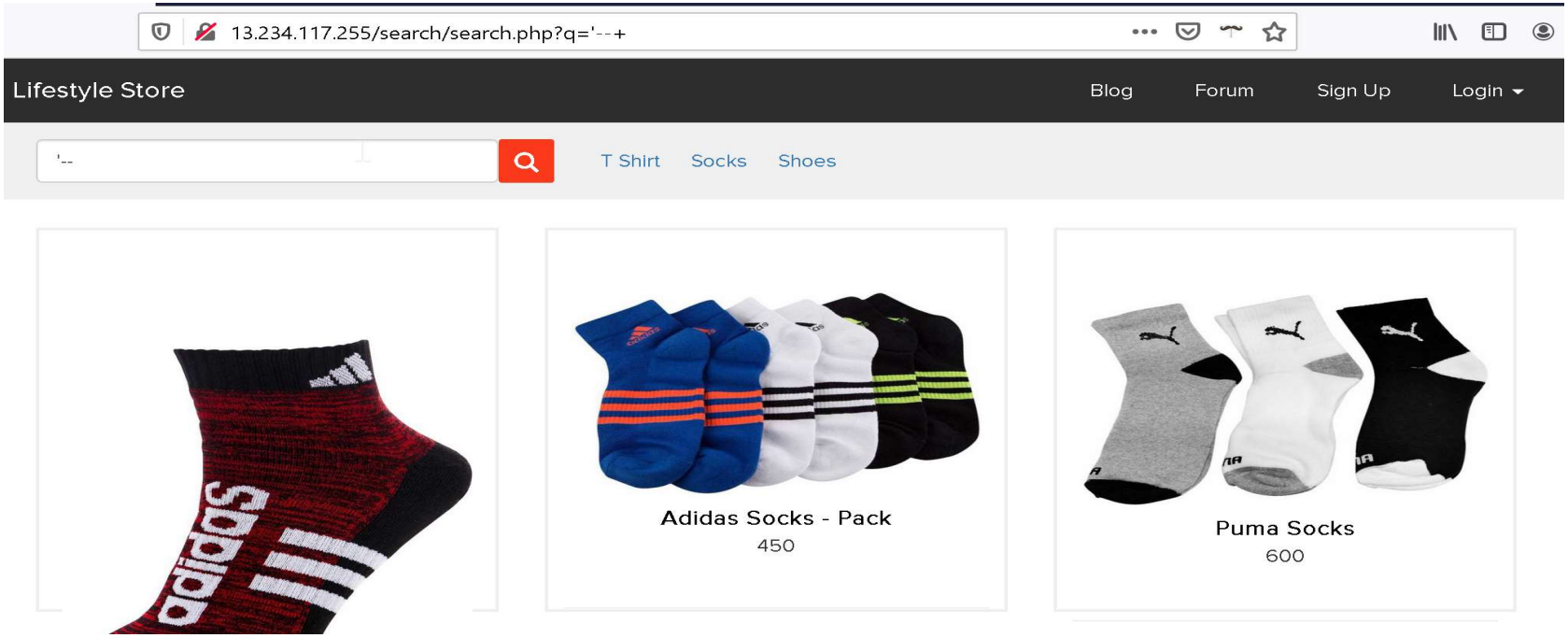
Clicking on shop now button,we get a search option on the top left corner,where we can search for anything on the website,as-



Observation

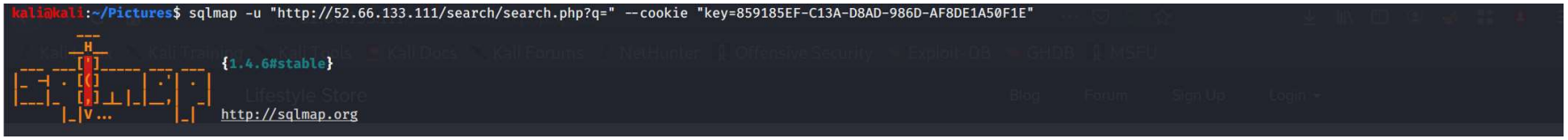
We apply single quote at the end of q parameter and we get complete MySQL error:

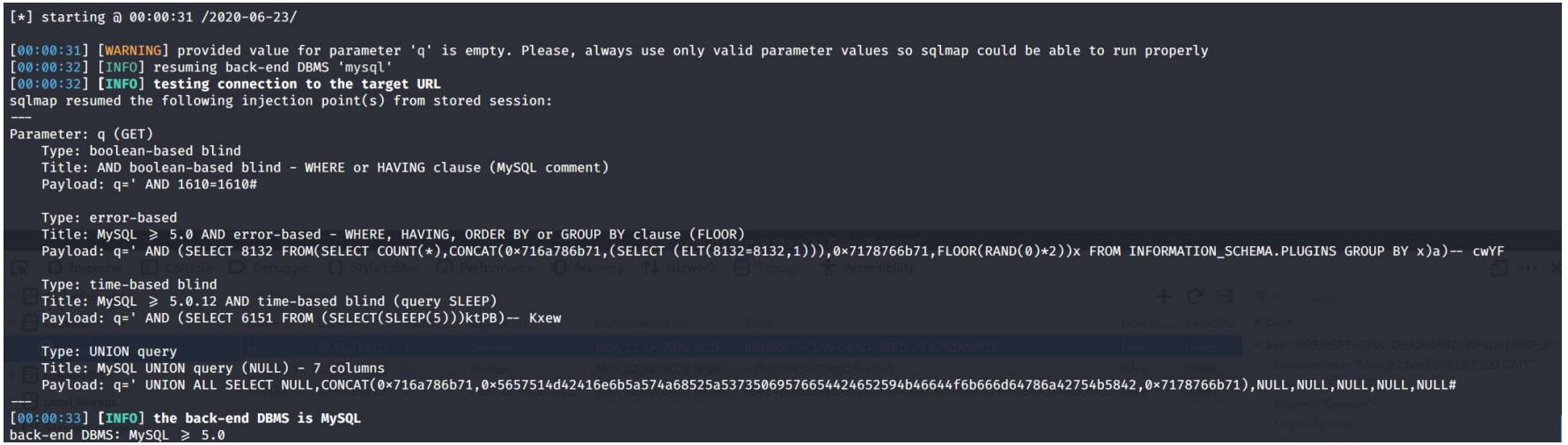
Also, we can the try q='--+ in the URL and the result confirmed sql injection.



Observation:

We used sqlmap for further advancing sql injections on the q parameter and got the result, both as shown:





Proof of Concept (PoC)

Here is the database names and few of its content extracted using sqlmap using --dbs switch at the end of the same command:

```
kali@kali:~/Pictures$ sqlmap -u "http://52.66.133.111/search/search.php?q=" --cookie "key=859185EF-C13A-D8AD-986D-AF8DE1A50F1E" --dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.4.6#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org
```

```
[00:02:34] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.0
[00:02:34] [INFO] fetching database names
[00:02:35] [WARNING] reflective value(s) found and filtering out
available databases [2]:
[*] hacking_training_project
[*] information_schema
```

Proof of Concept (PoC)

Even the 'cat' parameter is vulnerable to SQLi.

```
[*] starting @ 00:08:51 /2020-06-23/

[00:08:51] [WARNING] provided value for parameter 'cat' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[00:08:51] [INFO] resuming back-end DBMS 'mysql'
[00:08:51] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: cat=-5850' OR 7567=7567#

    Type: error-based
    Title: MySQL ≥ 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: cat=' OR (SELECT 6654 FROM(SELECT COUNT(*),CONCAT(0×7162717871,(SELECT (ELT(6654=6654,1))),0×7171767a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- uqUo

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=' AND (SELECT 2551 FROM (SELECT(SLEEP(5)))coaX)-- KoTa

    Type: UNION query
    Title: MySQL UNION query (NULL) - 7 columns
    Payload: cat=' UNION ALL SELECT NULL,NULL,CONCAT(0×7162717871,0×6a696b6671575548487847765 45a61766b697976416844496f41646c706a6679544c5a69654e5847,0×7171767a71),NULL,NULL,NULL,NULL#
---
[00:08:52] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.0
```

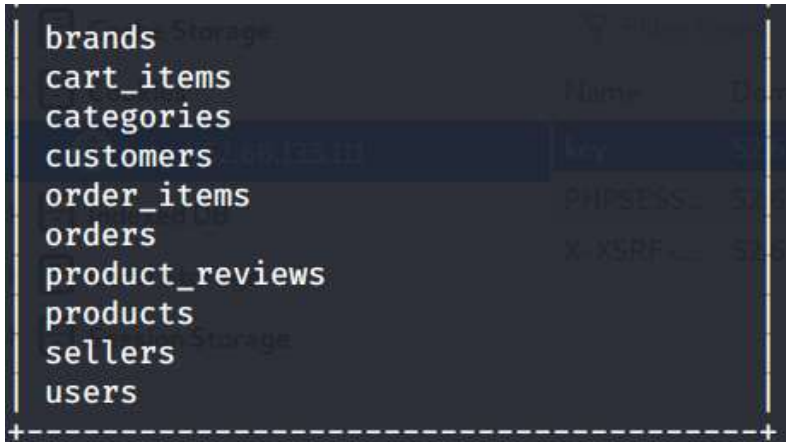
Proof of Concept (PoC)

- No of databases: 2
  - information_schema
  - hacking_training_project

- No of tables in information_schema: 61

  Few of them are:
  - FILES
  - EVENTS
  - COLUMNS
  - ENGINES
  - GLOBAL_STATUS
  - GLOBAL_VARIABLES
  - COLUMN_PRIVILEGES

No. of tables in hacking_training_project:10



Business Impact – High

Using this vulnerability, attacker can execute arbitrary SQL commands on Websites server and gain complete access to internal databases along with all customer data inside it.

Alongside is the screenshot of users table which shows user credentials being leaked ,although with hashing,dictionary-based and logic-based bruteforcing passwords still remains an option to verify with.

user_name	type	password	phone_number
admin	admin	$2y$10$Phrdr2F1sC912mG6jY5af.Qbdj7O6yasyHc/CZiNEchBPsWJiWuK2	8521479630
Donal234	customer	$2y$10$PM.7nBSP5FMaldXiM/S3s./p5xR6GTKvjry7ysJtxOkBqOJURAHsO	9489625136
Pluto98	customer	$2y$10$ba4bpp3nqfFRPB9.w.s4KeU36ecbRemyM6bj65FI/Q1Et0Qv1X9QK	8912345670
chandan	seller	$2y$10$4cZBEIrgthXdvT1hwUlivuFELe03rR.GIcdp03NjrlS0veiOKLVDa	7854126395
Popeye786	customer	$2y$10$Fkv1RfwYTioW0w2CaZtAQuXVnhGAUjt/If/yTqkNPC5zTrsVm7EeC	9745612300
Radhika	seller	$2y$10$RYxNhOyV/G4g7OtFwpqYaexvHi8rF6XXui8kT1WtrfqhTutCA8JC.	9512300052
Nandan	seller	$2y$10$G.cRNLMEiG79ZFXElHg.R.o95334U0xmZu4.9MqzR5614ucwnk59K	7845129630
MurthyAdapa	customer	$2y$10$mzQGzD4sDSj2EunpCioe4eK18clAbs0T2P1a1P6eV1DPR.11UubDG	8365738264
john	customer	$2y$10$GhDB8h1X6XjPMY12GZ1vDO7Y3en97u1/.oXTZLmYqB6F18FBgecvG	6598325015
bob	customer	$2y$10$kiUikn3HPFbuyTtK75lLNurxzqC0LX3eMGy0/Ux16JOoG37dCGKLq	8576308560
jack	customer	$2y$10$z/nyNlkRJ76m9ItMZ4N5lOeRxy6Gkqi9N/UBcJu5ZeO7eM7N4pTHu	9848478231
bulla	customer	$2y$10$HT5oiRMetqaZ7xGZPE9s2.Mk1yF4PnYDJHCWbm2w/xuKpjEEI/zjG	7645835473
hunter	customer	$2y$10$pB3U9iFxwBgSbl2AkBpiEeIBdhiYfWy9y.xV23q12gGbMCyn7N3g2	9788777777
asd	customer	$2y$10$At5pFZnRWpjCD/yNnJWDL.L3Cc4Cv0W8Q/WEHmWzBFqVIkBQFpCF2	9876543210
acdc	customer	$2y$10$J50B78.gpucuLTwpHwbcPedYcain.Yi.tsTLyQtK17FzdSpmIRRbi	9999999999
FindMe	customer	$2y$10$ieLZsBhtXY0N92Wyo3o5y.BQJO4zd7tpcF18XV61F/FhyBT6.zfNa	9999999999

Recommendation

Take the following precautions to avoid exploitation of SQL injections:

- Whitelist User Input: Whitelist all user input for expected data only. For example if you are expecting a flower name, limit it to alphabets only upto 20 characters in length. If you are expecting some ID, restrict it to numbers only
- Prepared Statements: Use SQL prepared statements available in all web development languages and frameworks to avoid attacker being able to modify SQL query
- Character encoding: If you are taking input that requires you to accept special characters, encode it. Example. Convert all **‘ to \’** , **“ to \”**, **\ to \\.** It is also suggested to follow a standard encoding for all special characters such has HTML encoding, URL encoding etc
- Do not store passwords in plain text. Convert them to hashes using SHA1 SHA256 Blowfish etc
- Do not run Database Service as admin/root user
- Disable/remove default accounts, passwords and databases
- Assign each Database user only the required permissions and not all permissions

References

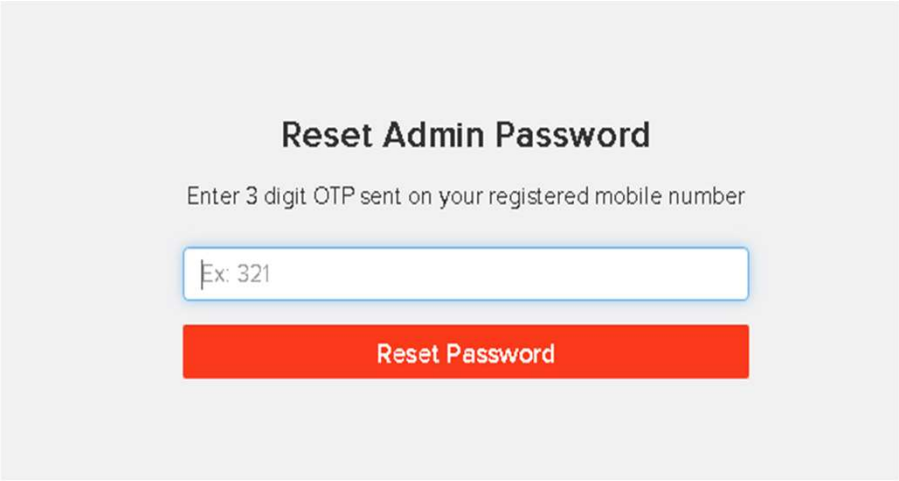
- *https://www.owasp.org/index.php/SQL_Injection*
- *https://en.wikipedia.org/wiki/SQL_injection*

2.Account takeover via OTP Bypass

Account Takeover Using OTP Bypass (Critical)	The below mentioned login page allows login via OTP which can be brute forced. **Affected URL :** • http://52.66.133.111/reset_password/admin.php **Affected Parameters :** • OTP (POST parameters)

Observation:

- When we navigate to admin login page i.e. 'http://52.66.133.111/login/admin.php', and try to reset the password, we are asked for the three digit OTP as:



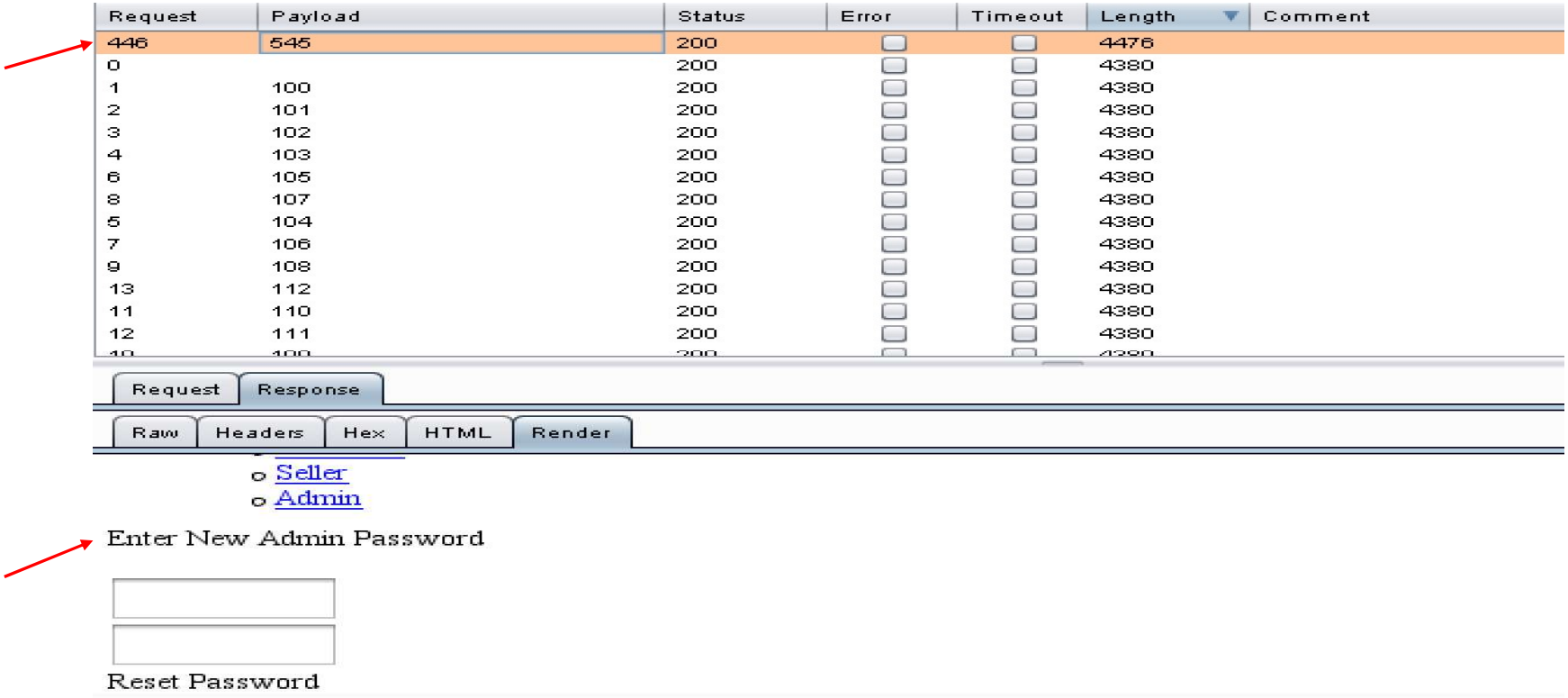
Observation:

Following request will be generated containing OTP parameter.

```
GET /reset_password/admin.php?otp=321 HTTP/1.1
Host: 52.66.133.111
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://52.66.133.111/reset_password/admin.php
Cookie: key=859185EF-C13A-D8AD-986D-AF8DE1A50F1E; PHPSESSID=duc88d90rv3kmvid3e5a6gtk11;
X-XSRF-TOKEN=5351c1a1b15d9a2b69826304b7fb0a3df592ffa940c2446d762163a9c1eaff51
Upgrade-Insecure-Requests: 1
```

Observation

We shoot the request with all possible combinations of 3 Digit OTPs and upon a successful hit, we get a response containing “Enter New Admin Password”. We can use the same OTP then to login.



Request	Payload	Status	Error	Timeout	Length	Comment
446	545	200	☐	☐	4476	
0		200	☐	☐	4380	
1	100	200	☐	☐	4380	
2	101	200	☐	☐	4380	
3	102	200	☐	☐	4380	
4	103	200	☐	☐	4380	
6	105	200	☐	☐	4380	
8	107	200	☐	☐	4380	
5	104	200	☐	☐	4380	
7	106	200	☐	☐	4380	
9	108	200	☐	☐	4380	
13	112	200	☐	☐	4380	
11	110	200	☐	☐	4380	
12	111	200	☐	☐	4380	

Business Impact: Extremely High

After logging in as admin, the attacker gets almost all the control of the website business.

Right from changing the seller details with their corresponding product to adding false product. They can make customers pay for false products and this may have a great business impact and hamper the privacy too.

Also ,changing cost of products can cause severe financial loss to the owner .

Recommendation:

1)The length of the OTP should be done at least 6. This makes bruteforcing impractical.

2)There should be at least two-step verification before resetting password.

3)Captcha can be used to protect from bruteforcing.

4)Number of attempts can be limited.

References:

1)https://www.cloudways.com/blog/what-is-brute-force-attack/

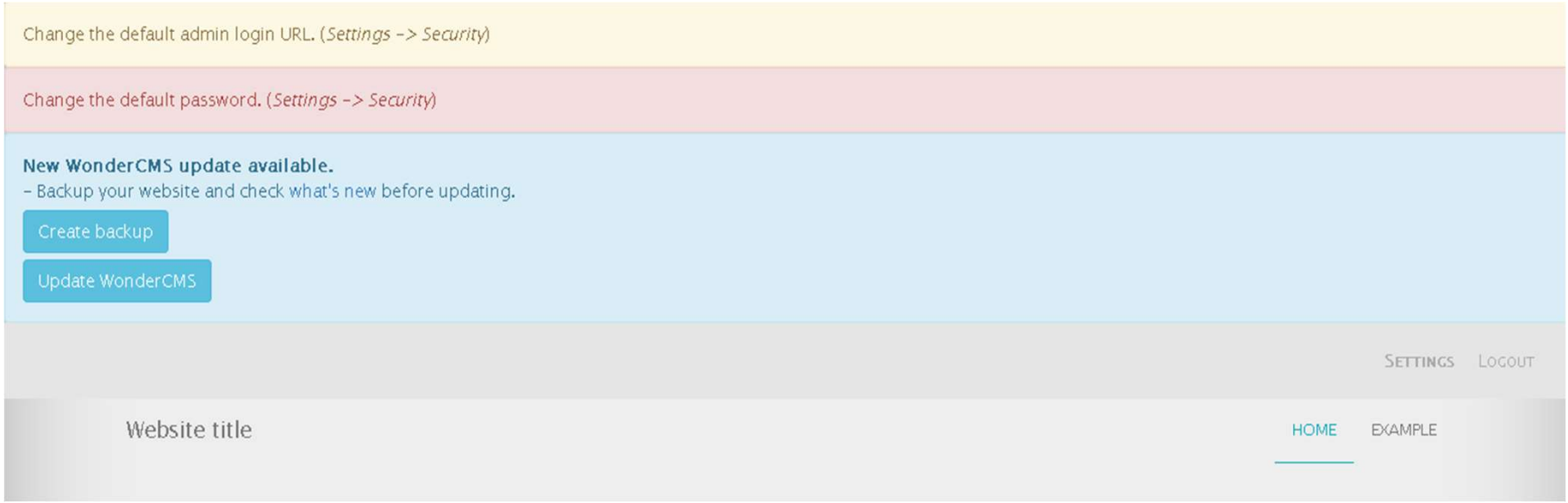
2)https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks

3.Access to admin panel :

Access to admin panel (Critical)	Below mentioned URL can be used to access admin panel of website. **Affected URL :** • http://52.66.198.61/wondercms/loginURL

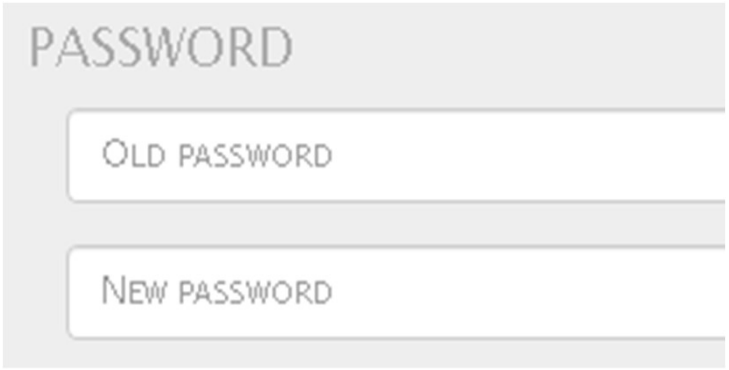
Observation:

When we open 'http://52.66.198.61/wondercms/ ' url ,we get the password on the page and login as admin in the url 'http://52.66.198.61/wondercms/loginURL' :



Proof Of Concept(PoC):

We can change password making the actual admin unable to login the next time. We can also add and delete pages:



Business Impact(Extremely High):

- Using this vulnerability, the attacker can get complete access to the blog of the website.
- The attacker can change the password or even change the url of the admin panel and restrict the admin to access it.
- Even pages can be created and deleted along with editing.
- Files can be added (without verification) and hence can be dangerous to the entire website, as the control of the entire website can be taken.

Recommendation:

- The default password should be changed and a strong password must be setup.
- The admin URL must also be such that its not accessible to normal users.
- Password changing option must be done with 2 to 3 step verification.
- Password must be at least 8 characters long containing numbers, alphanumeric, etc.
- All the default accounts should be removed.
- Password should not be reused.

References:

1. https://www.owasp.org/index.php/Testing_for_weak_password_change_or_reset_functionalities_(OTG-AUTHN-009)

2. https://www.owasp.org/index.php/Default_Passwords

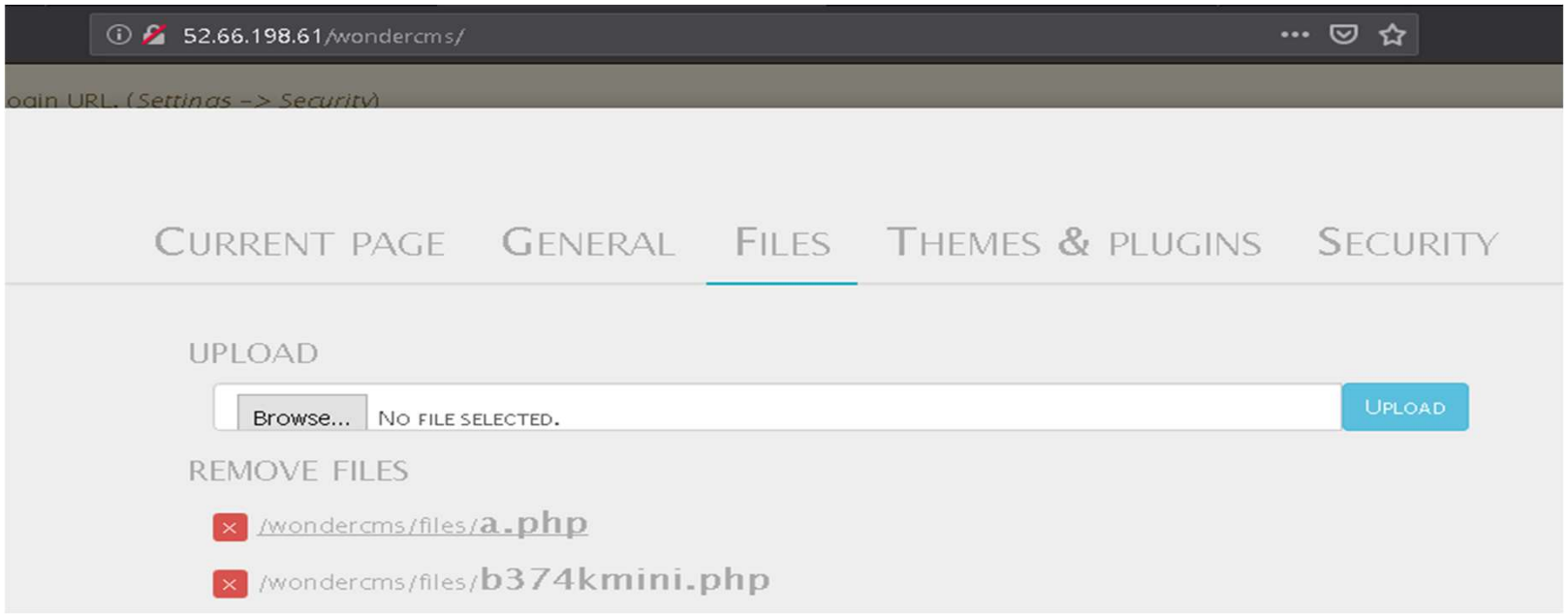
3. https://www.us-cert.gov/ncas/alerts/TA13-175A

4. Arbitrary File Upload Vulnerability

Arbitrary File Upload (Critical)	Below mentioned URL is vulnerable to Arbitrary File Upload **Affected URL :** • http://52.66.198.61/wondercms/ **File Uploaded**: 1.Sample php file 2.B374kmini.php(php shell)

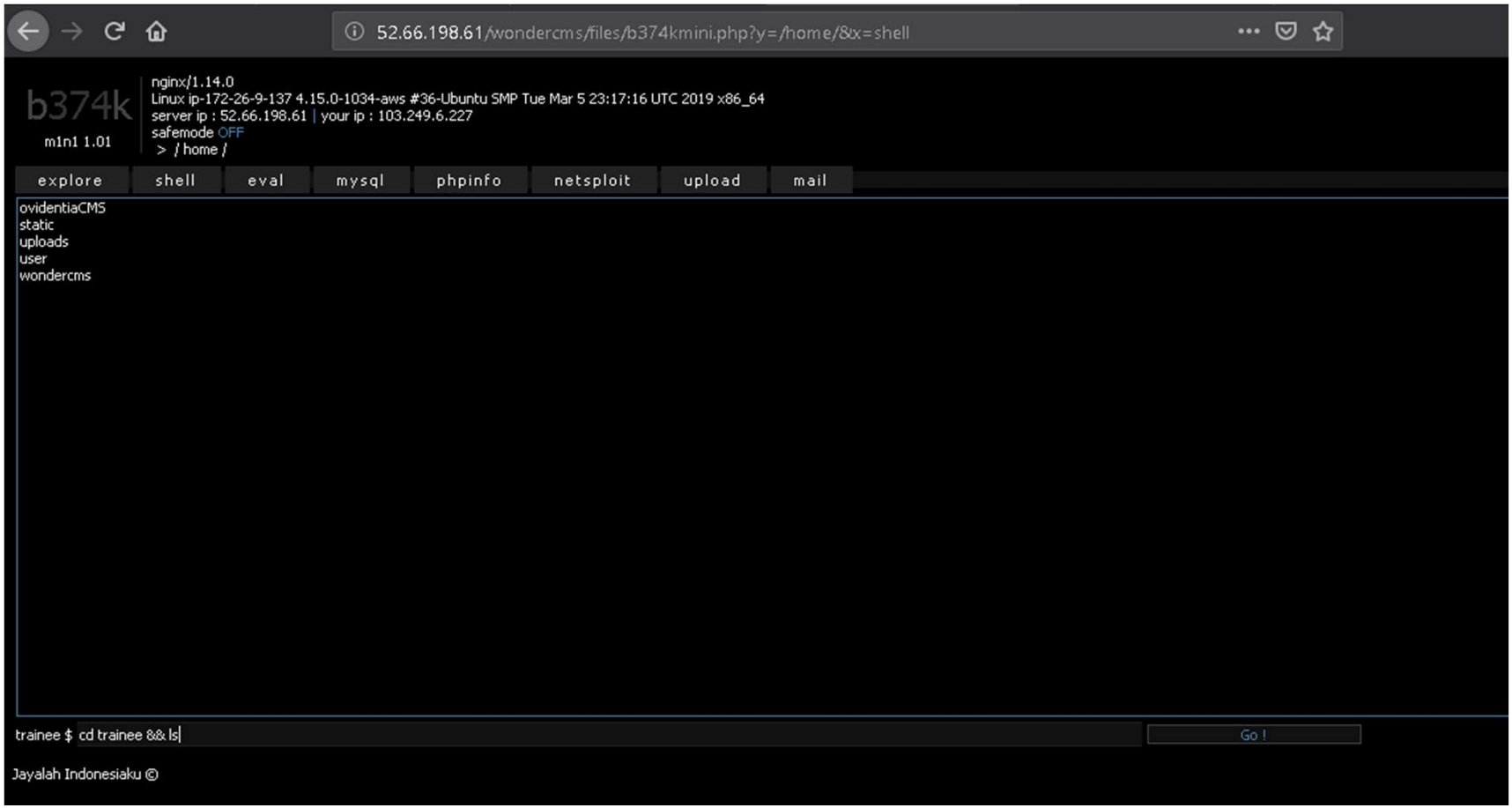
Observation:

When we click on blog option, and login to the admin panel we see file upload option in settings, where we can upload *any* kind of file.



Proof of Concept(PoC):

When we click on the mini shell which we uploaded,we get the acces to directories by typing commands in the shell option and much more .



Business Impact(High):

Using this vulnerability, the attacker can get the complete control of the system, database and can also forward attacks in the back-end systems along with client-side attacks or simple defacement.

The impact of this vulnerability is high, supposed code can be executed in the server context or on the client side. A malicious file such as a Unix shell script, a windows virus, an Excel file with a dangerous formula, or a reverse shell can be uploaded on the server in order to execute code by an administrator or webmaster later -- on the victim's machine.

An attacker might be able to put a phishing page into the website or deface the website and much more.

Recommendations:

- Blacklisting File Extensions like .php,.php5,.shtml,etc
- Whitelist the file extensions like .jpg,.pdf,etc
- Content type header validation
- Using File type detector
- Perform proper server-side validations on what kind of a file user is uploading
- Use white lists filters instead of black list filters. Example: in case of a resume upload feature, instead of banning PHP and .exe files, only allow .pdf, .doc and .docx files
- Rename the files using a code, so that the attacker cannot play around with file names
- Use static file hosting servers like CDNs and File Clouds to store files instead of storing them on the application server itself

References:

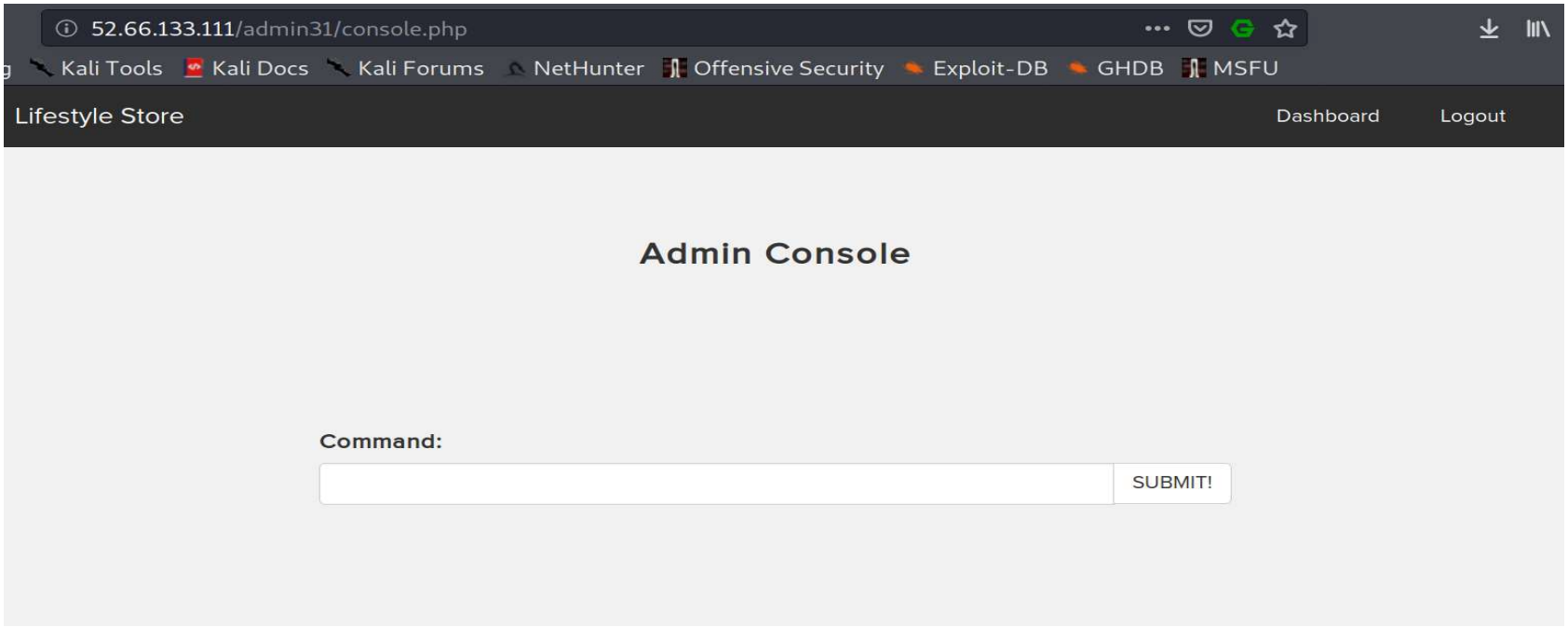
- https://www.owasp.org/index.php/Unrestricted_File_Upload
- https://www.opswat.com/blog/file-upload-protection-best-practices

5. Command Execution Vulnerability:

Command Execution Vulnerability (Critical)	Below mentioned URLs is vulnerable to Command Execution **Affected URL :** • http://52.66.198.61/admin31/console.php • http://52.66.198.61/wondercms/files/b374kmini.php?y=/home/trainee/wondercms/files/&x=shell (to directly to console of the php shell)

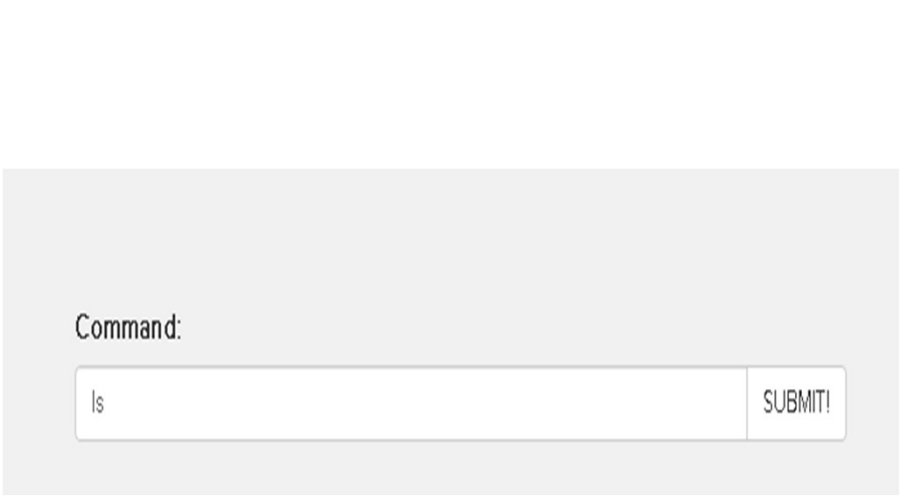
Observation:

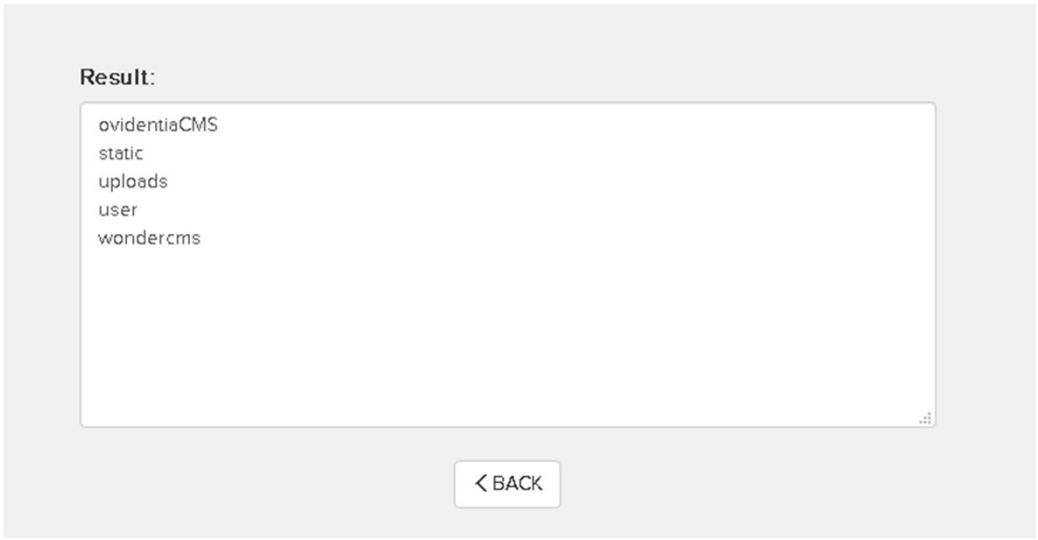
After logging in as admin, when we navigated to the URL 'http://52.66.198.61/admin31/console.php', we got the command line page where we can execute any command.



Proof Of Concept(PoC):

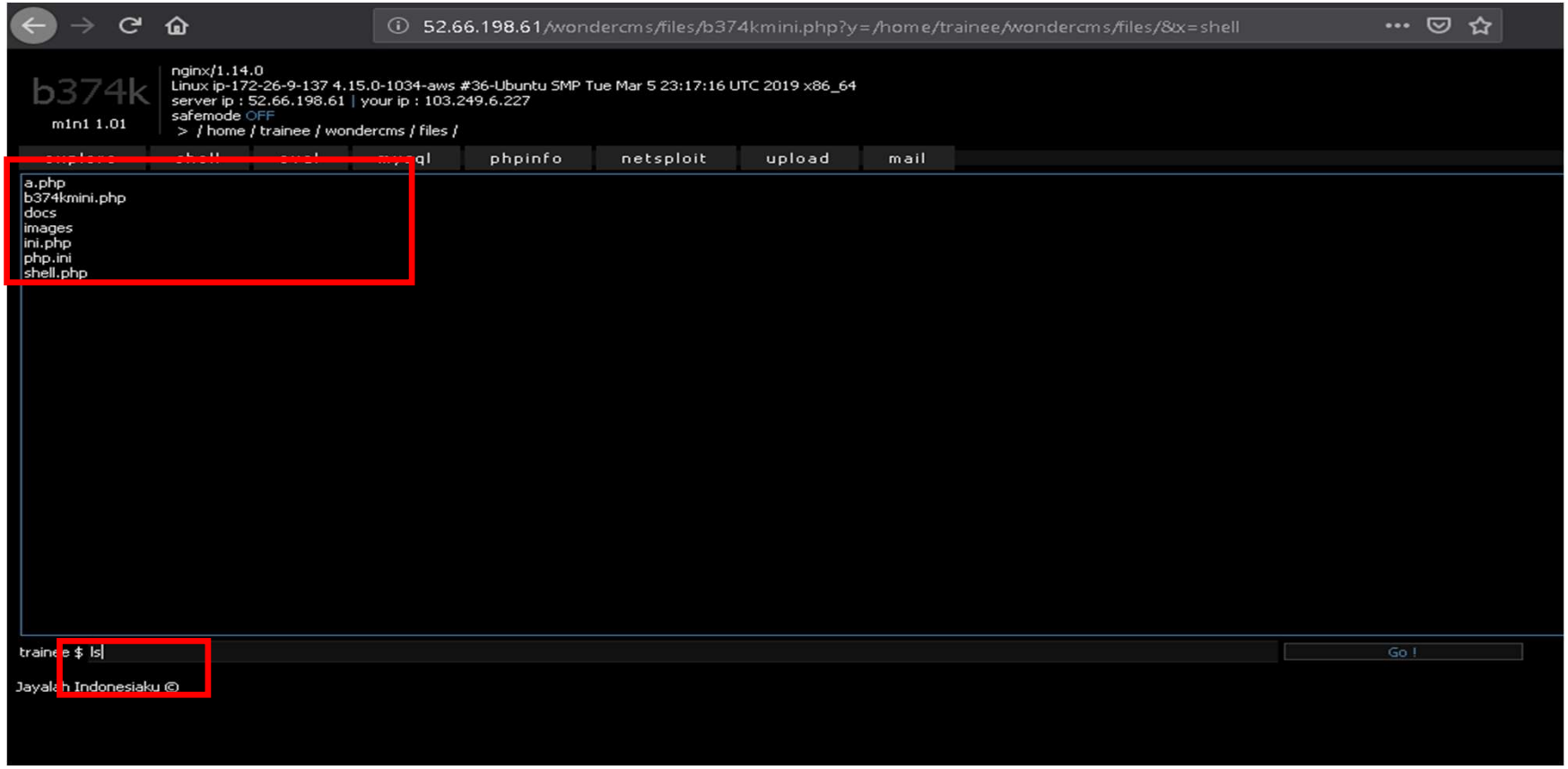
When we typed ls ,the result was as follows:





Proof Of Concept(PoC):

When we opened the b274k mini shell(after uploading it) in admin settings of the blog at URL 'http://52.66.198.61/wondercms/', and after clicking on shell option:



Business Impact(High):

- If the attacker enters into the admin account and finally to the console url,the he can put in any malicious code to extract or even edit data,as he the has the admin privileges.
- Other than entering malicious code and stuffs, the attacker can even get the details of the websites and its components like its version and hence find vulnerabilities to exploit them.
- If successfully exploited, impact could cover loss of confidentiality, loss of integrity, loss of availability, and/or loss of accountability.

Recommendations:

1)There should be filters so that malicious code cannot be injected in .

2)Input validation can be done.

3)Output Validation can be done.

4)Canonicalization can also be done.

References:

1)https://www.owasp.org/index.php/Command_Injection

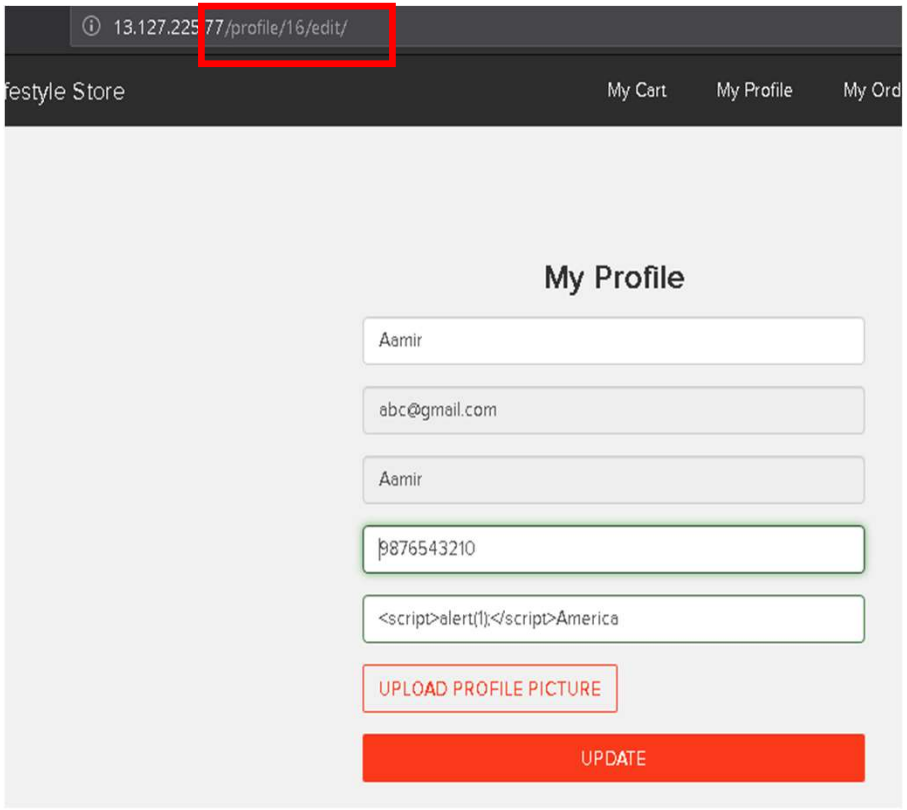
2)https://www.owasp.org/index.php/Code_Injection

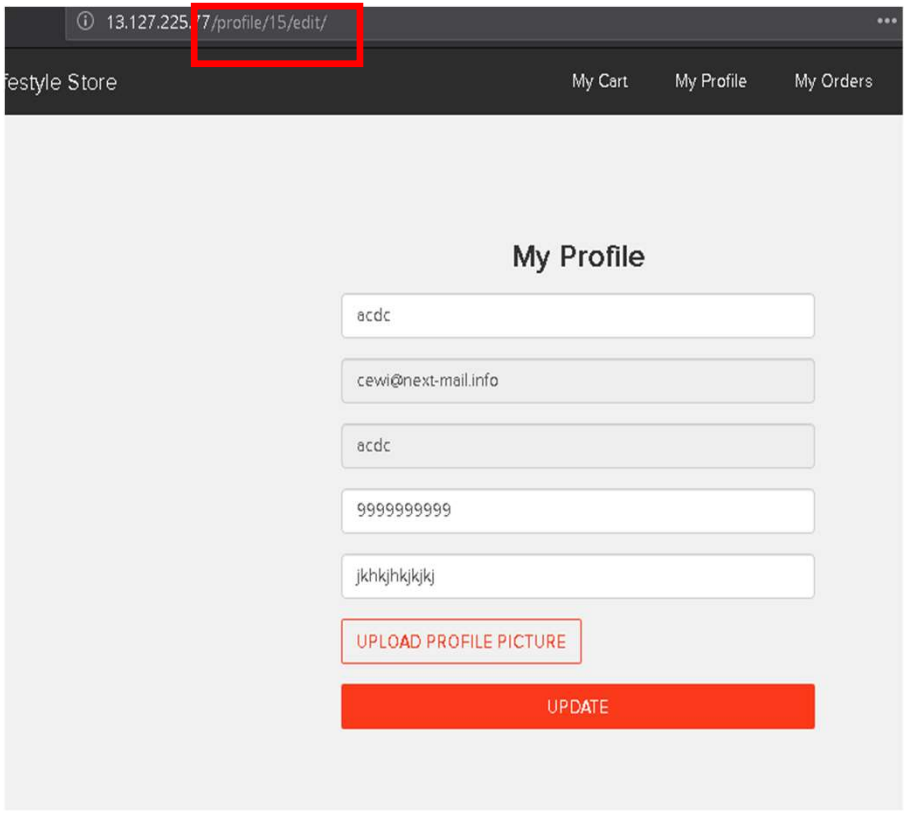
5. Unauthorised access to customer details using IDOR + Rate limiting flaw

Unauthorised access to customer details (Critical)	Below mentioned URL is vulnerable to IDOR(Displaying customer details) **Affected URL :** • http://52.66.198.61/profile/16/edit/ • http://52.66.198.61/orders/orders.php?customer=16 **Affected Parameters :** • user_id(POST parameter) • customer(GET parameter) **Payload:** • Changing user_id from 16 to 15 in the url itself

Observation:

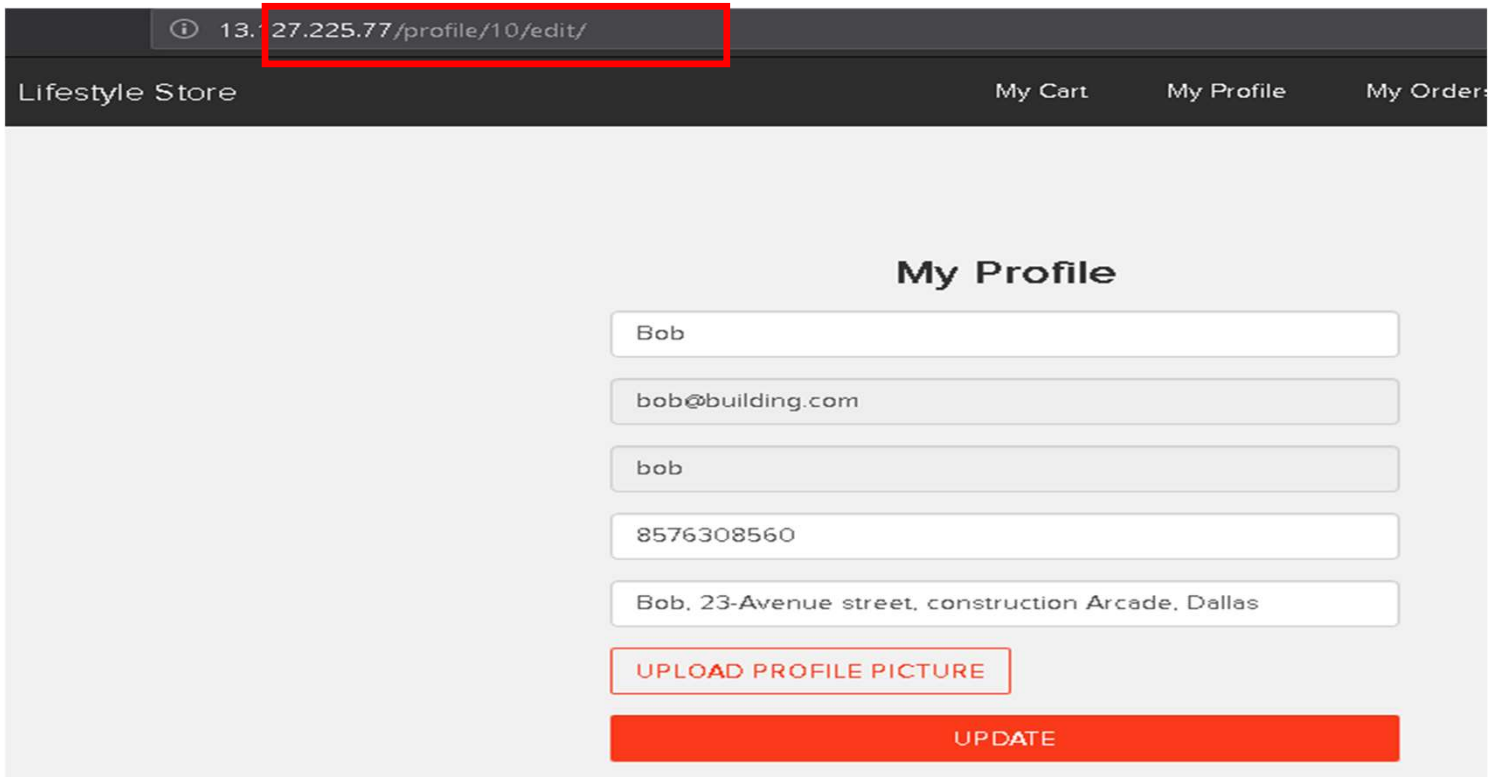
When the user_id was changed from 16 to 15, we got the entire details of user_id 15:





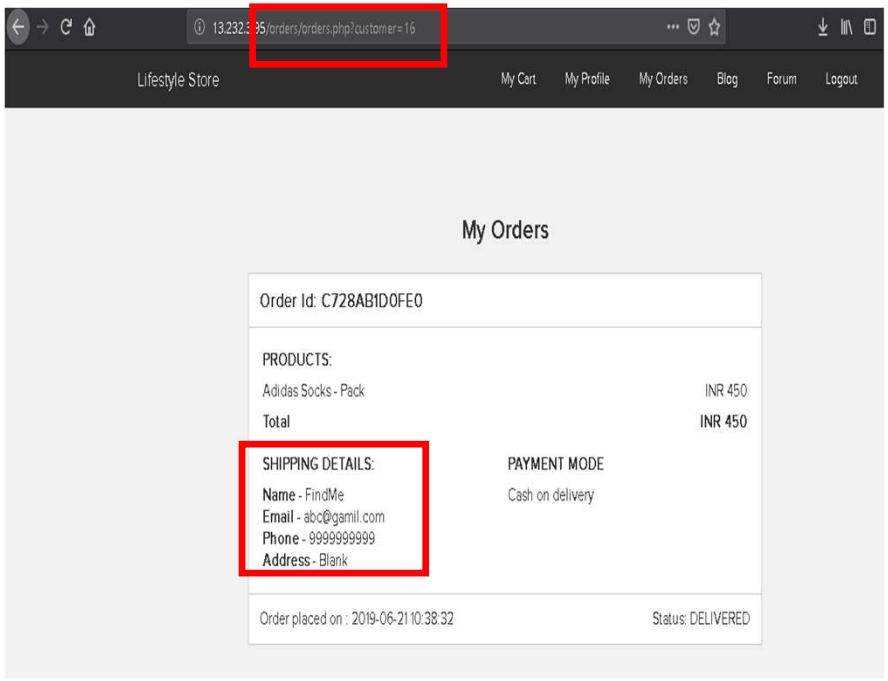
Proof Of Concept(PoC):

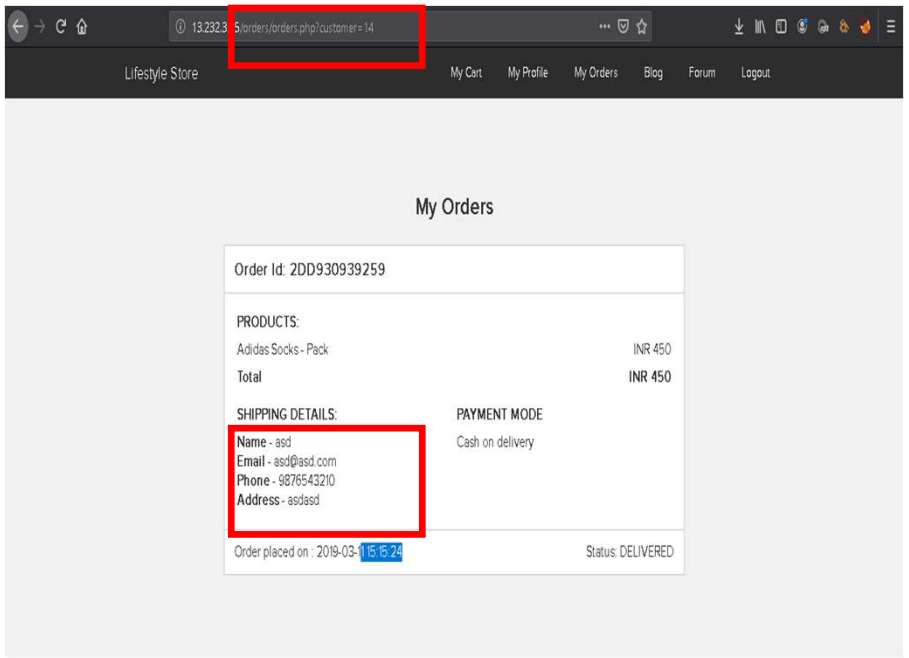
Even users with user_id 10 and others are accessible in editing section. Here, We can use burp suite intruder to bruteforce details of all the customers.{refer to Poc_5.mp4}



Proof Of Concept(PoC):

We can even look at the orders of other customers, other than the logged in account. Here, We can use burp suite intruder to bruteforce details of all the customers.{refer to Poc_5.mp4}





Business Impact(High):

Taking the advantage of this vulnerability, if the attacker can get these sensitive data of multiple users, it would be become one step easier for them to login to their account after getting their username.

The revealed data includes(name, username, email, address, phone number, etc.) are sufficient for performing a full fledge social engineering attack on the user.

Recommendation:

Take the following precautions:

- Implement proper authentication and authorisation checks to make sure that the user has permission to the data he/she is requesting
- Use proper rate limiting checks on the number of request comes from a single user in a small amount of time
- Make sure each user can only see his/her data only.

References:

https://www.owasp.org/index.php/Insecure_Configuration_Management
https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

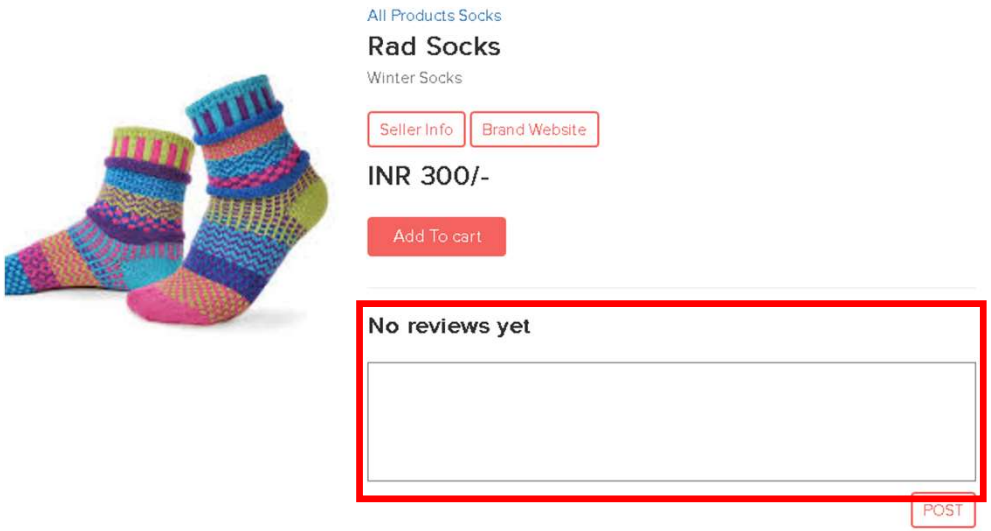
7)Cross Site Scripting(XSS):

Reflected Cross Site-Scripting (Severe)	Below mentioned parameters are vulnerable to Reflected XSS **Affected URL :** • http://13.233.132.83/search/search.php?q=qq **Affected Parameters :** • q (GET parameters) **Payload:** • “><script>alert(1);</script>

Stored Cross Site Scripting (Severe)	Below mentioned parameters are vulnerable to XSS **Affected URL :** • http://52.66.198.61/products/details.php?p_id=5 (all product ids have this vulnerability) **Affected Parameters :** • comment(POST parameters) **Payload:** • <script>alert(1);</script>
	Similar issue is found on below modules too **Affected URL :** • http://52.66.198.61/profile/16/edit/ **Affected Parameters :** • address(POST) **Payload:** • <script>alert(1);</script>
	Affected URL : • http://52.66.198.61/wondercms/ **Affected Parameters :** • title(POST),and all the below it. **Payload:** • <script>alert(1);</script>

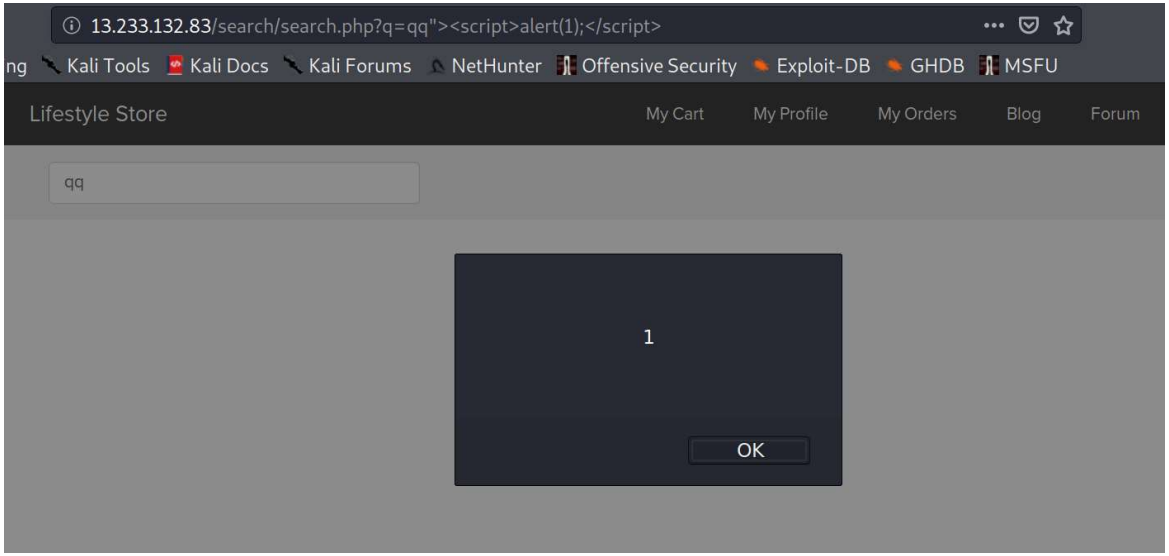
Observation:

When we navigate to 'http://52.66.198.61/products/details.php?p_id=5',we will see a review section below.



Proof Of Concept(Poc):

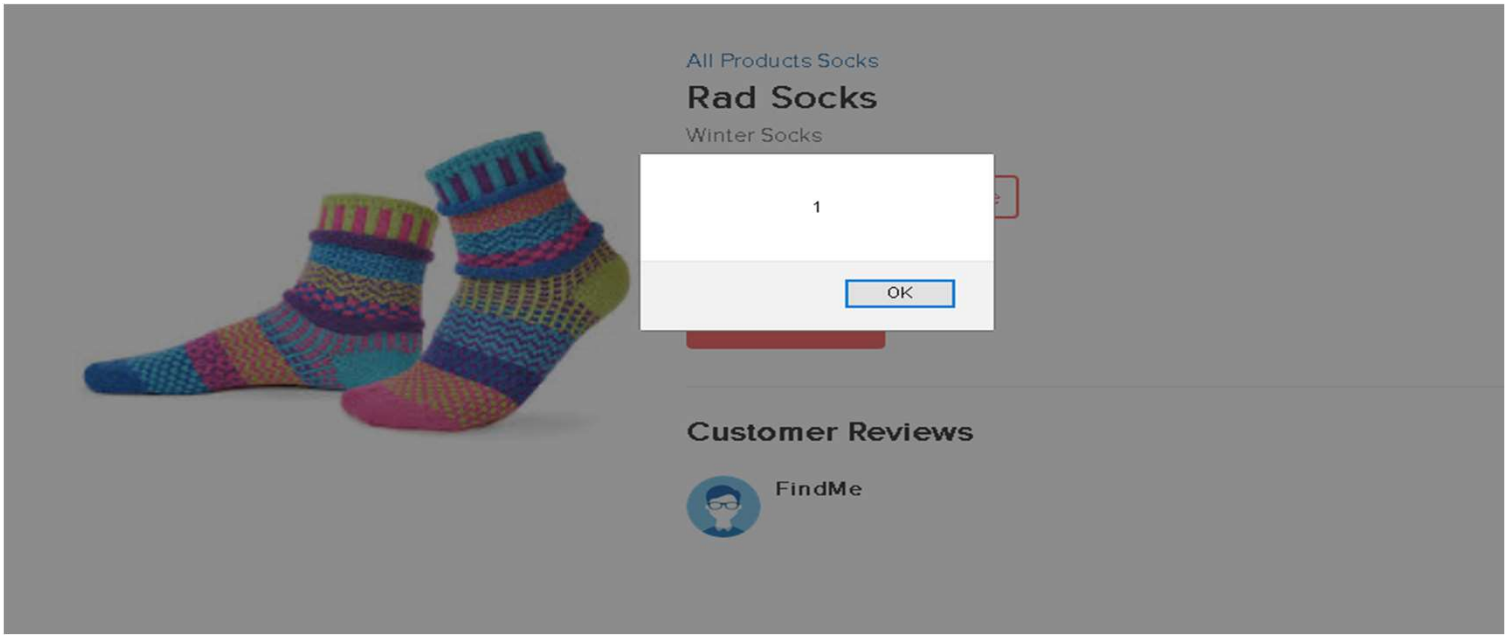
On visiting this URL http://13.233.132.83/search/search.php?q=qq%22%3E%3Cscript%3Ealert(1);%3C/script%3E ,
We get a prompt as:



Reflected Cross Site Scripting

Proof Of Concept(PoC):

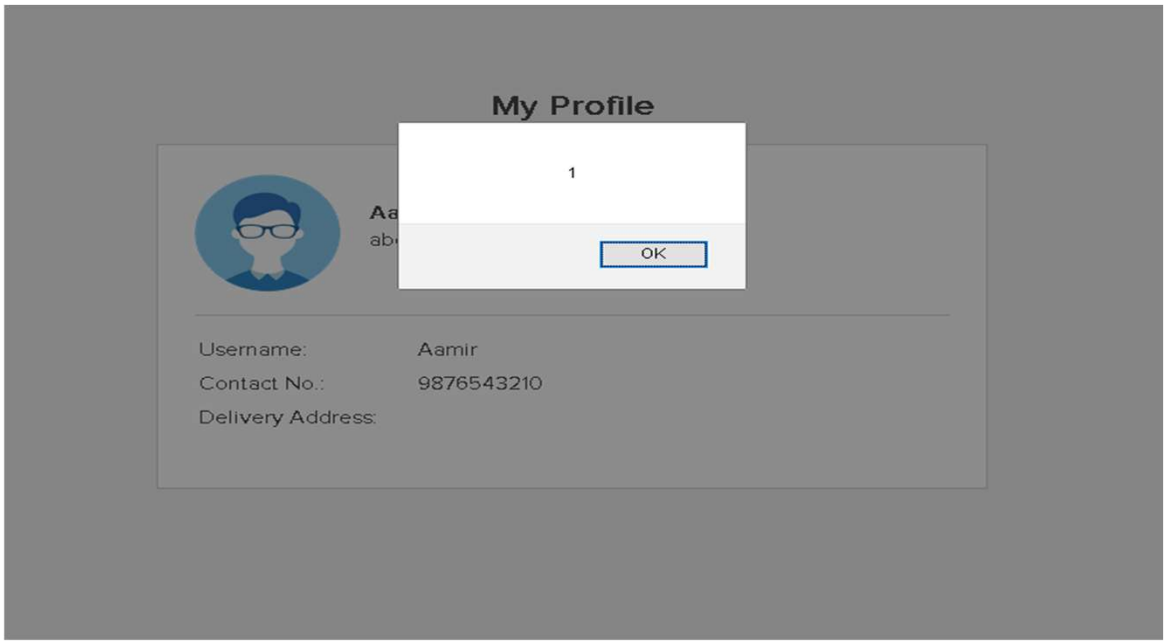
When we enter <script>alert(1);</script> and post it, we get the prompt as:



Stored Cross Site Scripting

Proof Of Concept(Poc):

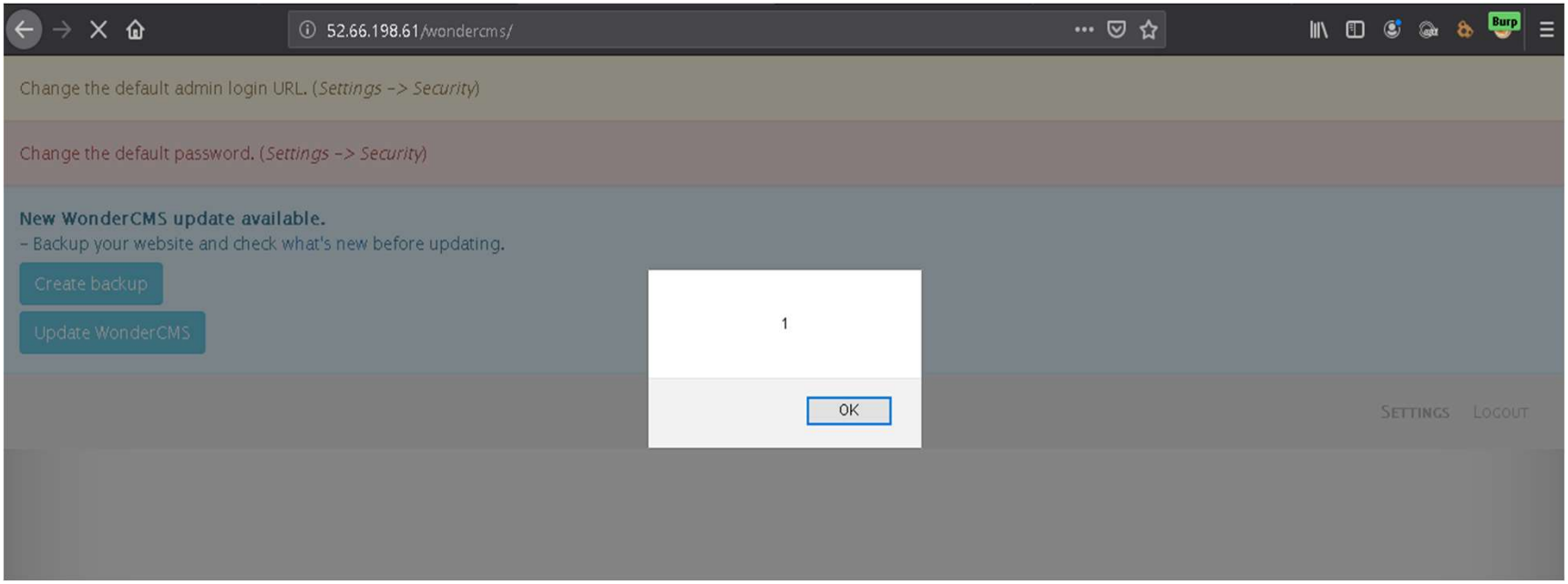
When we enter the same payload in the address portion and update it in the URL - 'http://52.66.198.61/profile/16/edit/' ,we get:



Stored Cross Site Scripting

Proof Of Concept(PoC):

When we add the same payload in the page title option in the settings of admin ,then we get the alert as:



Stored Cross Site Scripting

Business Impact – High

- As attacker can inject arbitrary HTML CSS and JS via the URL, attacker can put any content on the pa ge like phishing pages, install malware on victim's device and even host explicit content that could co mpromise the reputation of the organization

- All attacker needs to do is send the link with the payload to the victim and victim would see hacker c ontrolled content on the website. As the user trusts the website, he/she will trust the content.

Recommendation

Take the following precautions:

- Sanitise all user input and block characters you do not want
- Convert special HTML characters like ‘ “ < > into HTML entities " %22 < > before printing them on the website

References:

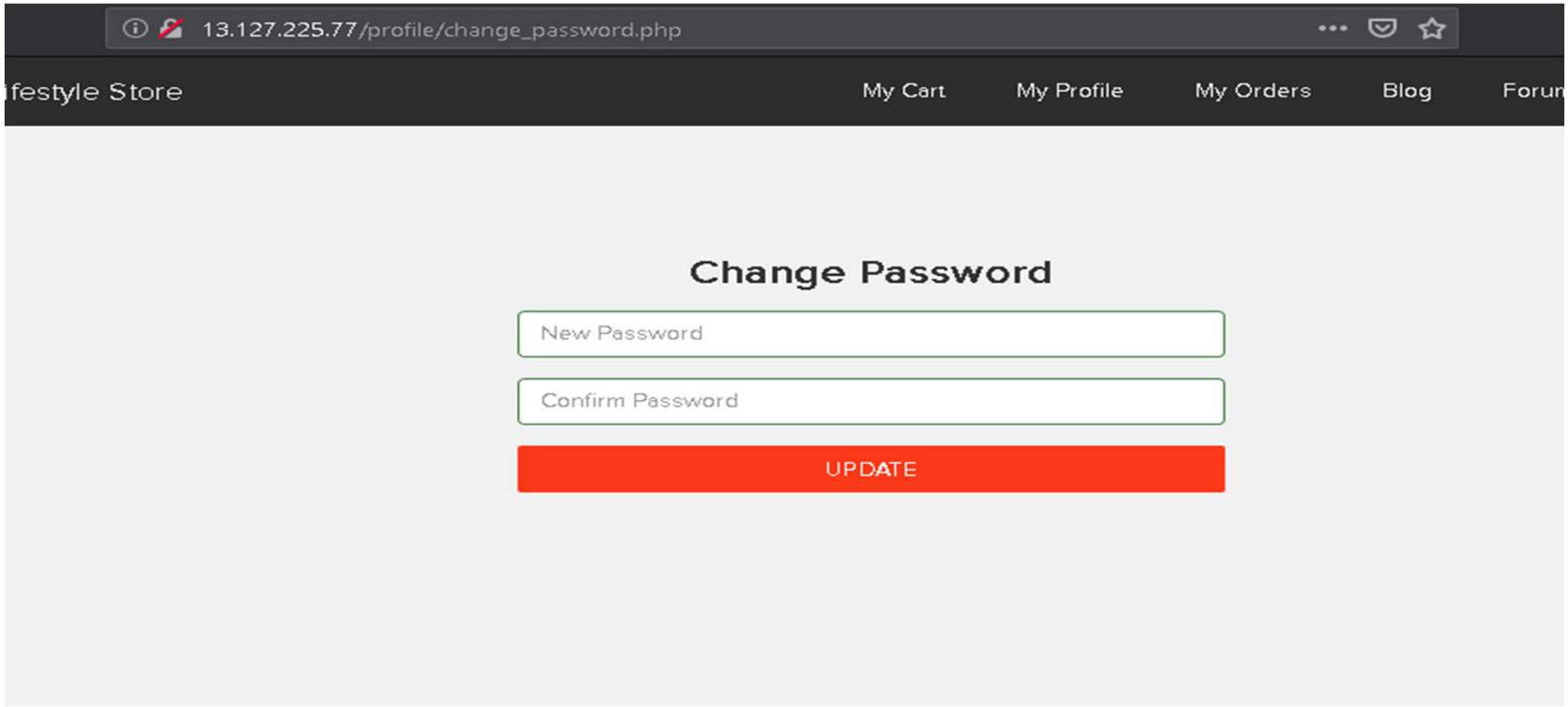
- *https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)*
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.w3schools.com/html/html_entities.asp

8)Cross Site Request Forgery(CSRF):

Cross Site Request Forgery(Severe)	CSRF is found in the modules below:- **Affected URL :** • http://52.66.198.61/profile/change_password.php **Affected Parameters :** • Update(POST)
	Affected URL : • http://52.66.198.61/cart/cart.php **Affected Parameters :** • Confirm order option(POST)

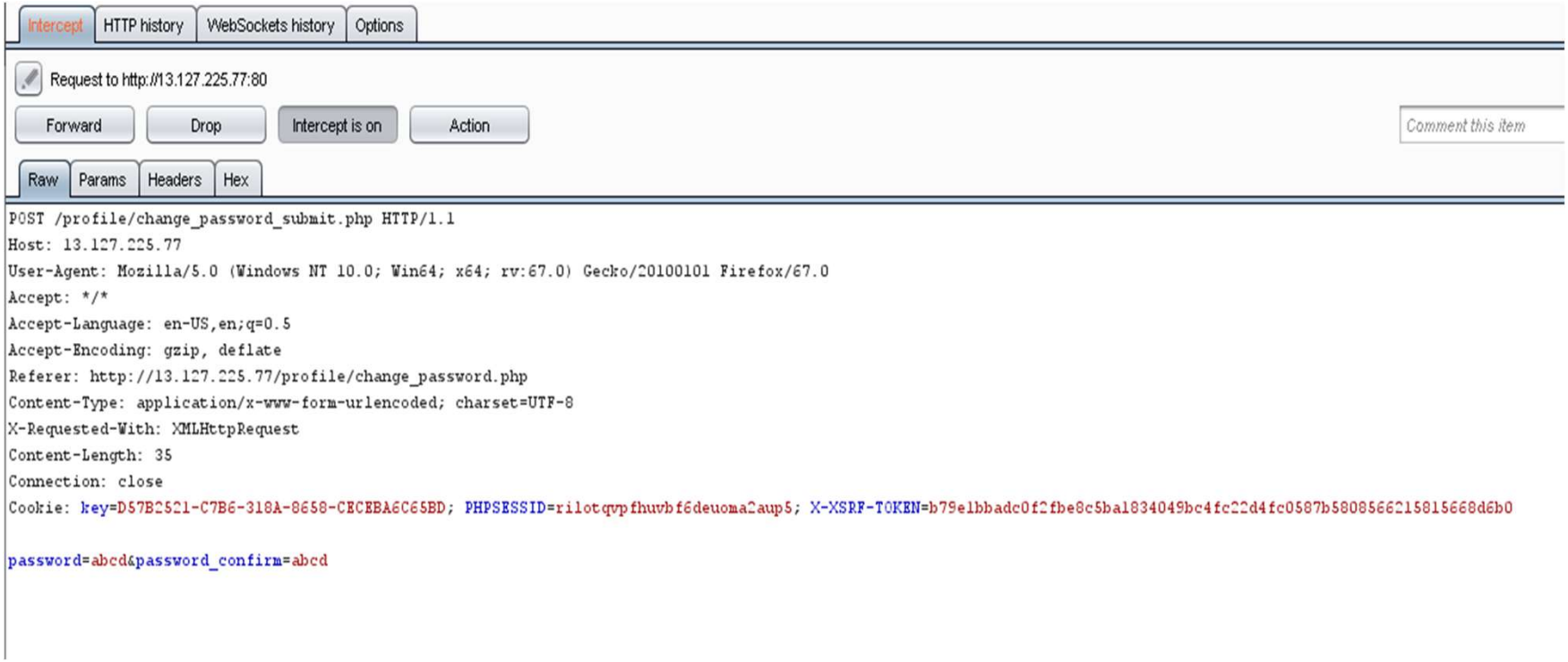
Observation:

We navigate to 'http://13.127.225.77/profile/change_password.php' after logging in to our account.



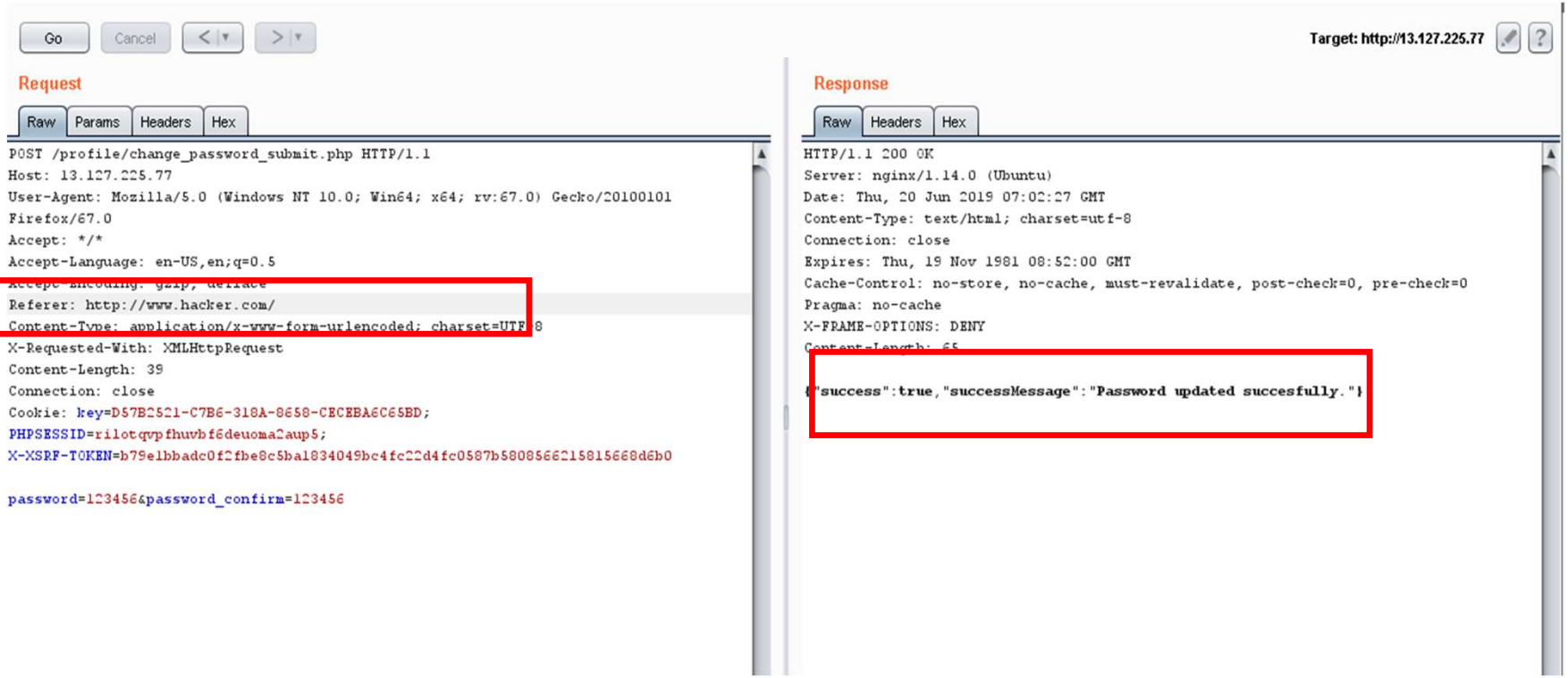
Observation:

To check that the update parameter checks the referrer or not ,we intercept it on burp suite, and send it to repeater for verification.



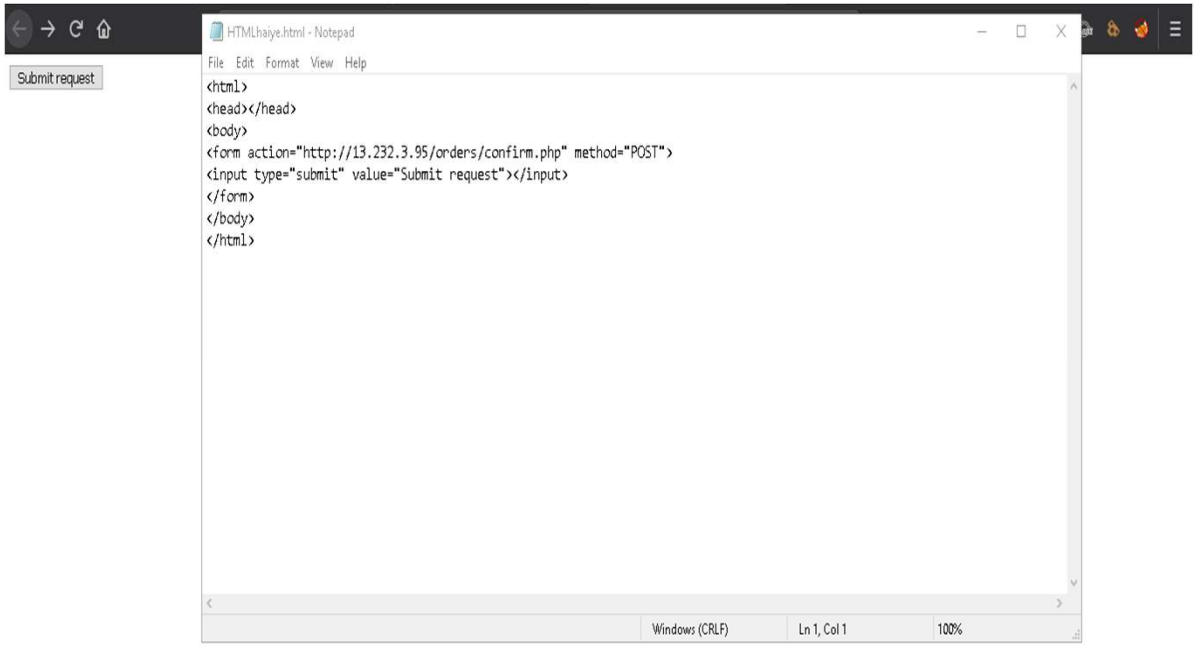
Proof Of Concept(Poc):

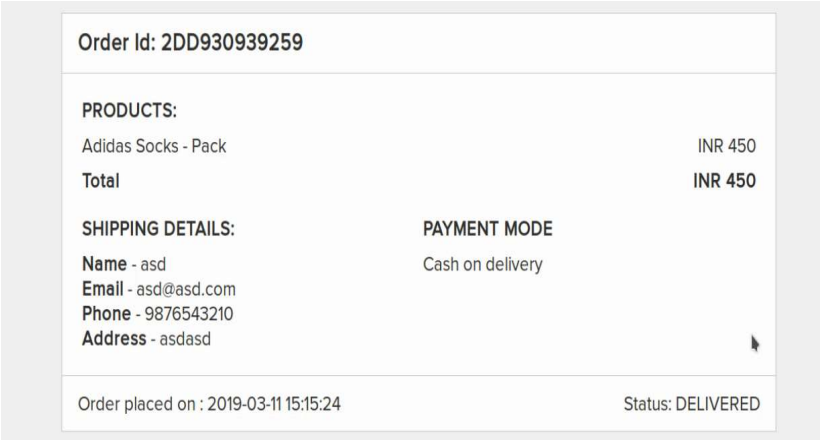
We change the referrer header in repeater and the password and forward the request,we see that it was successful in doing so,thus verifying CSRF.



Proof Of Concept(PoC):

After creating an html page with img source from confirm button on page 'http://13.232.3.95/cart/cart.php',when we run it and reload out order page,we see our cart as empty and order placed.





Business Impact (High):

If the attacker is able to exploit this vulnerability, the he may be able to order many items which the user would cancel later when the sender will send it to him(as its cash on delivery),so unnecessary load of workers can increase incredibly.

Moreover, the attacker maybe able to change the user's password without his consent, which in itself is highly insecure.

Recommendation:

- Ask the user his password (temporary like OTP or permanent like login password) at every critical action like while deleting account, making a transaction, changing the password etc.
- Implement the concept of CSRF tokens which attach a unique hidden password to every user in every
- Read the documentation related to the programming language and framework being used by your website
- Check the referrer before carrying out actions. This means that any action on x.com should check that the HTTP referrer is https://x.com/* and nothing else like https://x.com.hacker.com/*

References:

- https://en.wikipedia.org/wiki/Cross-site_request_forgery#Example_and_characteristics
- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

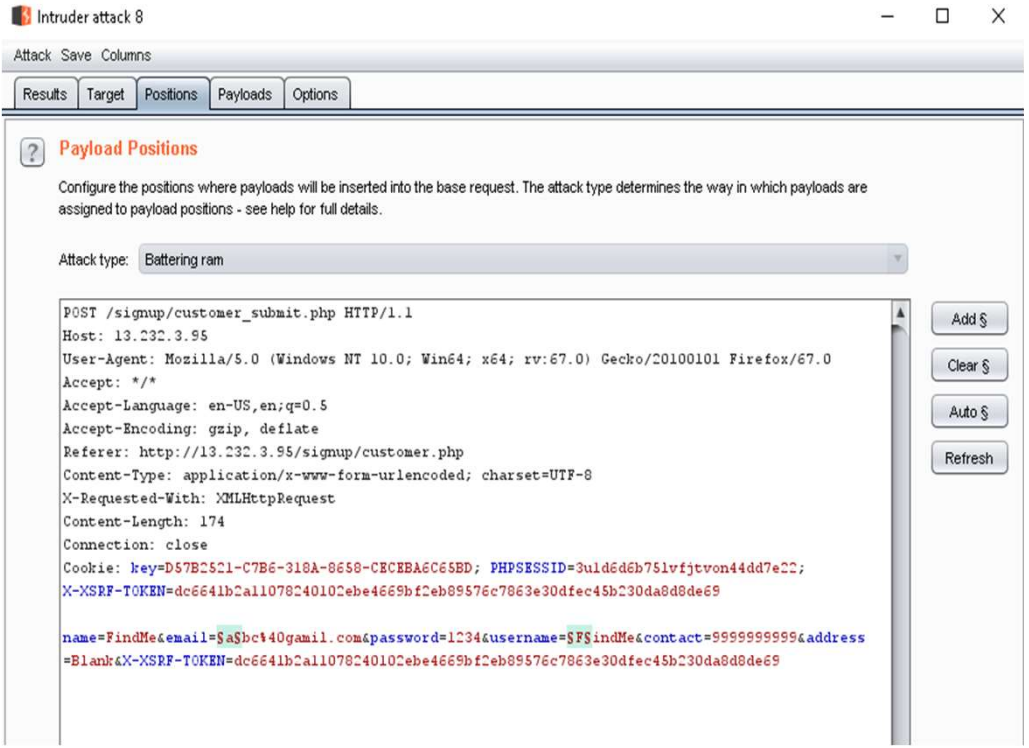
9. Rate Limiting Flaws:

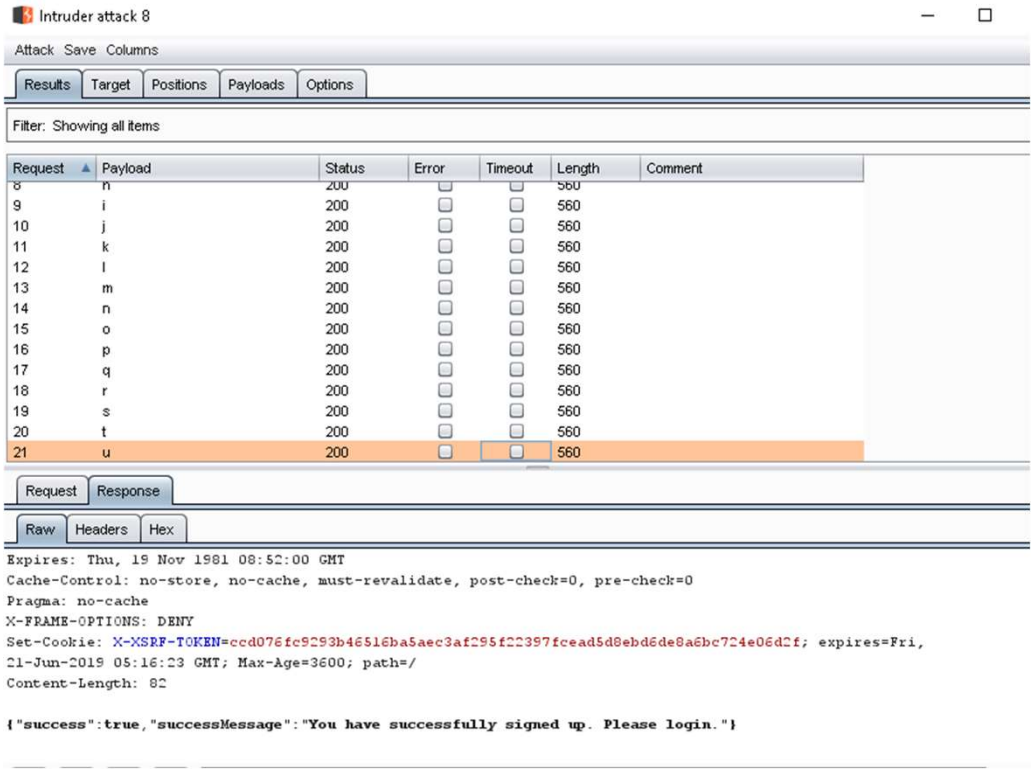
Rate Limiting Flaws(Severe)	Rate Limiting Flaws is found in the modules below:- **Affected URL :** • http://13.233.132.83/cart/cart.php **Affected Parameters :** • Coupan (Post parameter)
	Affected URL : • http://52.66.198.61/login/customer.php **Affected Parameters :** • Username and password
	Affected URL : • http://52.66.198.61/login/seller.php **Affected Parameters :** • Username and password
	Affected URL: • http://52.66.198.61/login/admin.php **Affected Parameters:** • Username and Password

Rate Limiting Flaws(Severe)	Rate Limiting Flaws is found in the following modules too:- **Affected URL :** • http://52.66.198.61/signup/customer.php
	Affected URL : • http://52.66.198.61/wondercms/loginURL **Affected Parameters :** • Password
	Affected URL: • http://52.66.198.61/forum/index.php?u=/user/login **Affected Parameters:** • Username and Password

Observation:

When we came to signup page of the lifestyle store, then after filling the details we intercepted it on burp suite, and sent to intruder to change the values of username and email, and hence we were able to successfully create many accounts simultaneously.





Observation:

Similarly ,we were able to intercept and try multiple attacks on all the listed login and signup sites of the website and its blog too after intercepting them on burp suite.

Proof Of Concept(Poc):

The POC Video is attached as: POC_9.mp4

Business Impact(High):

- If the attacker exploits this vulnerability, he may create a havoc on the webserver due to creation of a million accounts or a million attempts of login, which can further also lead to denial of service.
- Furthermore, this lack of rate limitation configuration may also help the attacker by fetching him more time to bruteforce and fetch passwords of users or even the admin.

Recommendation:

1)The length of the password should be large and strong(unrelated).This makes bruteforcing impractical.

2)There should be at least two-step verification before creating an account.

3)Captcha can be used to protect from bruteforcing.

4)Number of attempts can be limited.

References:

1)https://www.cloudways.com/blog/what-is-brute-force-attack/

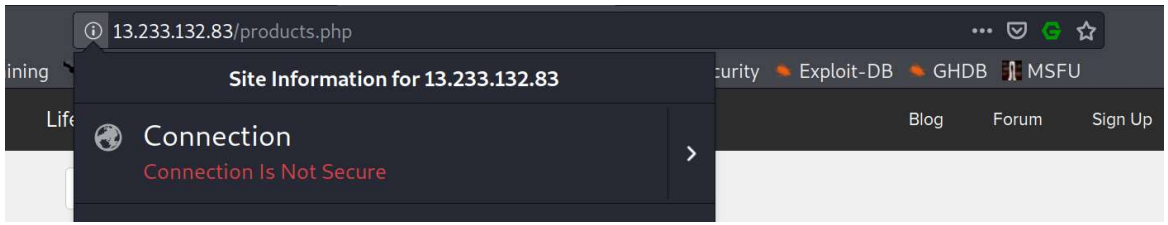
2)https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks

10)Crypto Configuration Flaw

Crypto Configuration Flaw(Severe)	Cryto Configuration Flaws are found in the modules below:- **Affected URL :** • http://52.66.198.61/ (All the webpages ,blogs,forum)

Observation:

Clearly ,all the webpages use 'http' and not 'https' which is far less secure and not encrypted. It is also indicated by Browser



Proof Of Concept(Poc):

Capture the network traffic through wireshark and see the transferred data in plaintext.

```
POST /signup/customer_submit.php HTTP/1.1
Host: 13.234.117.255
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 185
Origin: http://13.234.117.255
Connection: keep-alive
Referer: http://13.234.117.255/signup/customer.php
Cookie: key=859185EF-C13A-D8AD-986D-AF8DE1A50F1E; PHPSESSID=68t8mebusd5827agbt9qk1vp24; X-XSRF-
TOKEN=6a8ce4a0dc11001cf4b86997d699e6ffc174ddea4e47365816d753ce4e5c8781

name=Test&email=livaxex164%40wkernl.com&password=Test&username=Test&contact=6728732738&address=dsasadsadsda&X-
XSRF-TOKEN=6a8ce4a0dc11001cf4b86997d699e6ffc174ddea4e47365816d753ce4e5c8781HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Sun, 21 Jun 2020 18:54:12 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-FRAME-OPTIONS: DENY
Set-Cookie: X-XSRF-TOKEN=f8c55de08910bc84c2585b4dd45f18390e11715f0051c056efdac7c5d716d102; expires=Sun, 21-
Jun-2020 19:54:05 GMT; Max-Age=3600; path=/
Content-Encoding: gzip
```

Business Impact(High):

Security is almost halved in http providing easy man-in-the-middle attack and others which makes it easy for attacker to go through the data transmitted over the internet/

Recommendation:

Use https and not http as the protocol.

References:

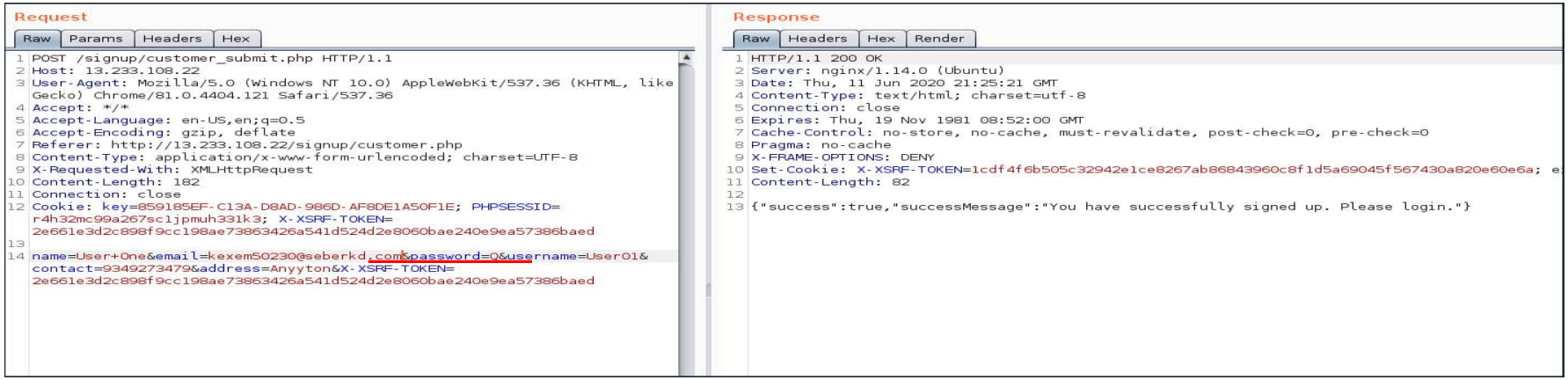
- https://www.owasp.org/index.php/Category:Cryptographic_Vulnerability
- https://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html

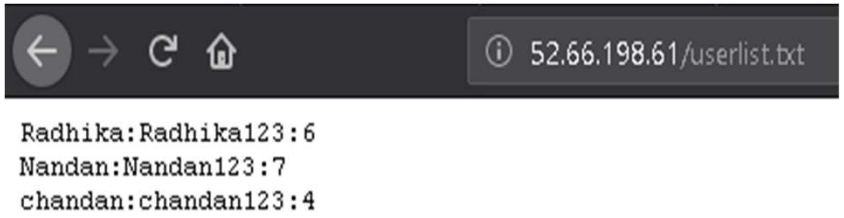
11. Weak Passwords Policy:

Crypto Configuration Flaw(Severe)	Cryto Configuration Flaws are found in the modules below:- **Affected URL :** • http://52.66.198.61/login/seller.php • http://52.66.198.61/wondercms/

Observation and POC:

The web application allows weak passwords to be used by customers, sellers, and admins. Here, only “Q ” can be used as a password.





Business Impact(High):

Easy, default and common passwords make it easy for attackers to gain access to their accounts illegal use of them and can harm the website to any extent after getting logged into privileged accounts.

Recommendation:

1. There should be password strength check at every creation of an account.
2. There must be a minimum of 8 characters long password with a mixture of numbers, alphanumeric, special characters, etc.
3. There should be no repetition of password, neither on change nor reset.
4. The password should not be stored on the web, rather should be hashed and stored.

References:

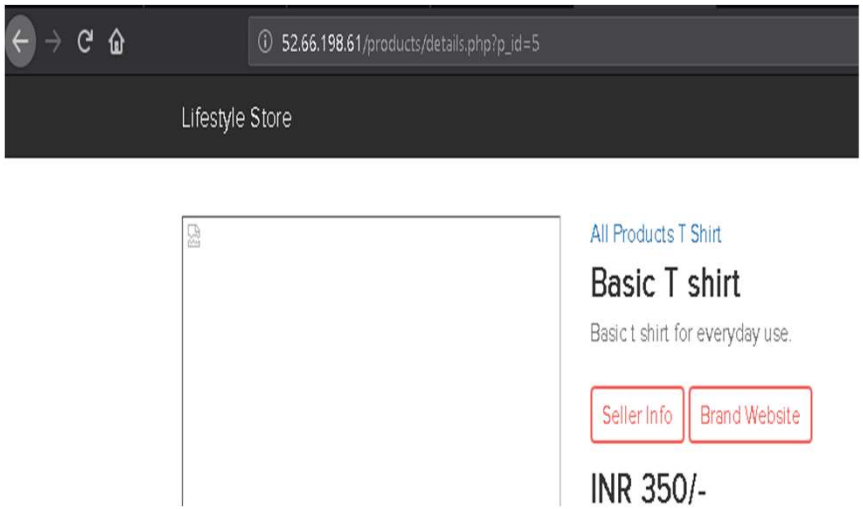
- https://www.acunetix.com/blog/articles/weak-password-vulnerability-common-think/
- https://www.owasp.org/index.php/Testing_for_Weak_password_policy_(OTG-AUTHN-007)

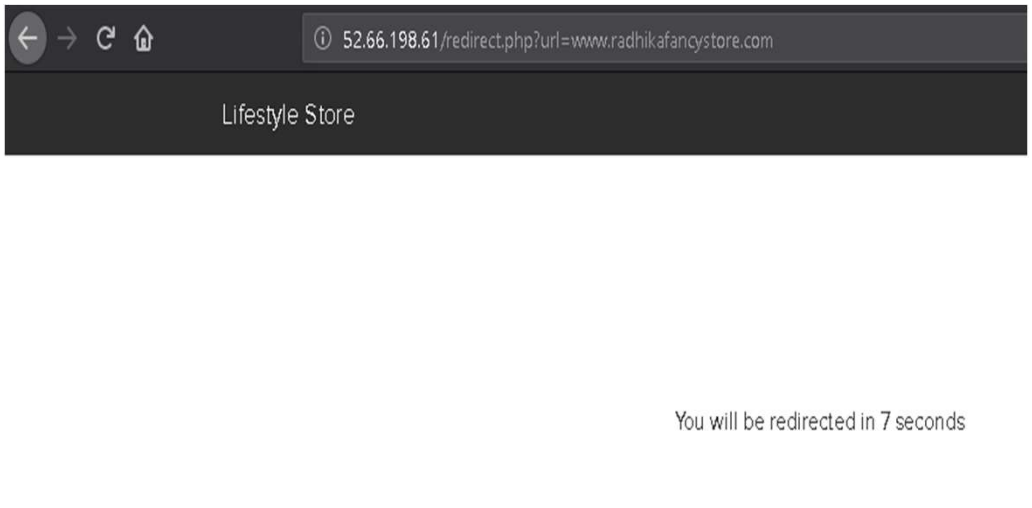
12. Open Redirection:

Open Redirection(Severe)	Open Redirection vulnerability are found in the module below:- **Affected URL :** • http://13.233.132.83/redirect.php?url=www.chandanstore.com **Affected parameter :** • url (Brand Website)

Observation:

When we view any product ,and click on “Brand Website” tab as shown, then we get redirected to it openly as GET parameter.





Proof Of Concept(Poc):

Just by changing the url parameter to any other site, we can redirect a user to any other site.

Ex:- http://13.233.132.83/redirect.php?url=www.google.com

Above link will redirect to Google page.

I have attached the POC video as: POC_12.avi

Business Impact(High):

If the attacker changes the URL to some malicious website looking similar to the given website, he can take the credentials and even credit card details on checkout from the user trust.

Recommendation:

If possible, applications should avoid incorporating user-controllable data into redirection targets. In many cases, this behavior can be avoided in two ways:

1. Remove the redirection function from the application, and replace links to it with direct links to the relevant target URLs.
2. Maintain a server-side list of all URLs that are permitted for redirection. Instead of passing the target URL as a parameter to the redirector, pass an index into this list.

If it is considered unavoidable for the redirection function to receive user-controllable input and incorporate this into the redirection target, one of the following measures should be used to minimize the risk of redirection attacks:

1. The application should use relative URLs in all of its redirects, and the redirection function should strictly validate that the URL received is a relative URL.
2. The application should use URLs relative to the web root for all of its redirects, and the redirection function should validate that the URL received starts with a slash character. It should then prepend http://yourdomainname.com to the URL before issuing the redirect.
3. The application should use absolute URLs for all of its redirects, and the redirection function should verify that the user-supplied URL begins with http://yourdomainname.com/ before issuing the redirect.

References:

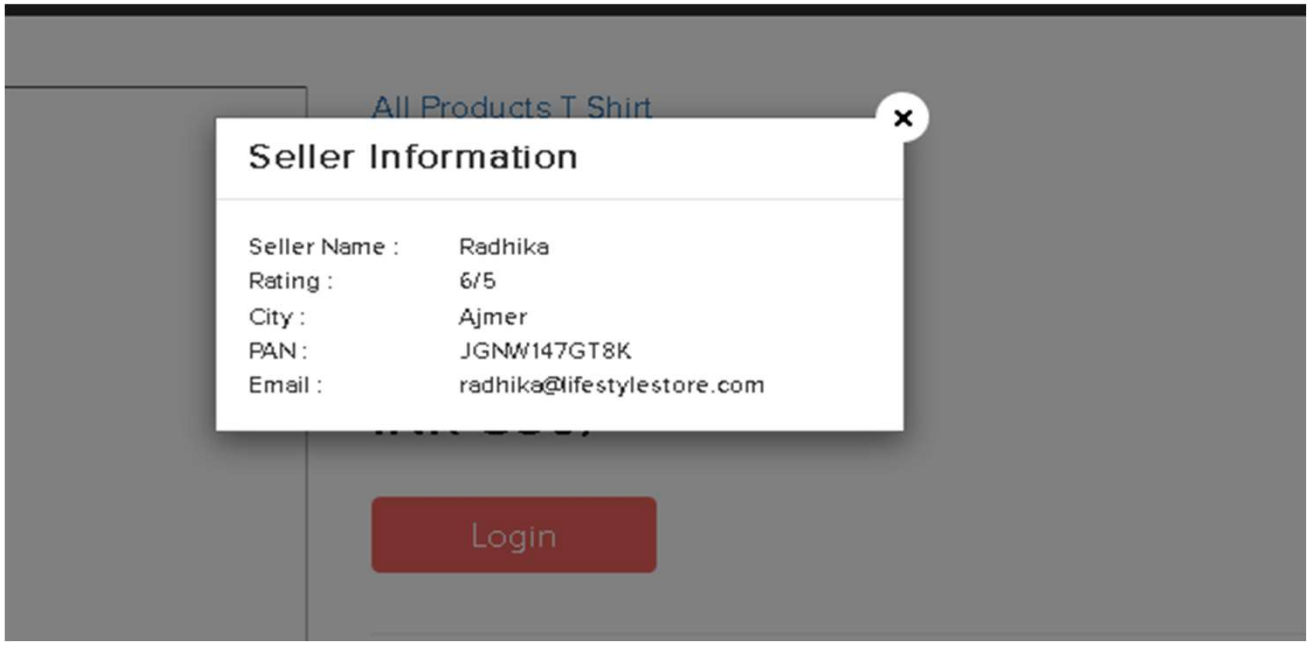
- https://portswigger.net/kb/issues/00500100_open-redirection-reflected
- https://www.owasp.org/index.php/Testing_for_Client_Side_URL_Redirect_(OTG-CLIENT-004)

13. PII Leakage:

Unrequired Details for seller (Moderate)	Below mentioned URL gives the unrequired details about the seller: **Affected URL :** • http://35.154.118.58/products/details.php?p_id=2(or any other p_id)

Observation:

When we click on the “Seller Info” option, we get the details of the seller ,even those which are not required like the pan card number.



Business Impact(Moderate):

There is no direct business impact in this case ,but this amount of information can definitely lead to social engineering attacks on the seller and can indirectly harm the business.

Recommendation:

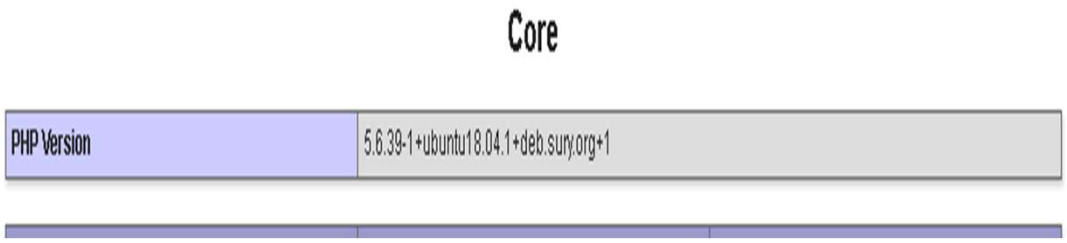
Only name and email is sufficient as far as the query or help is concerned.

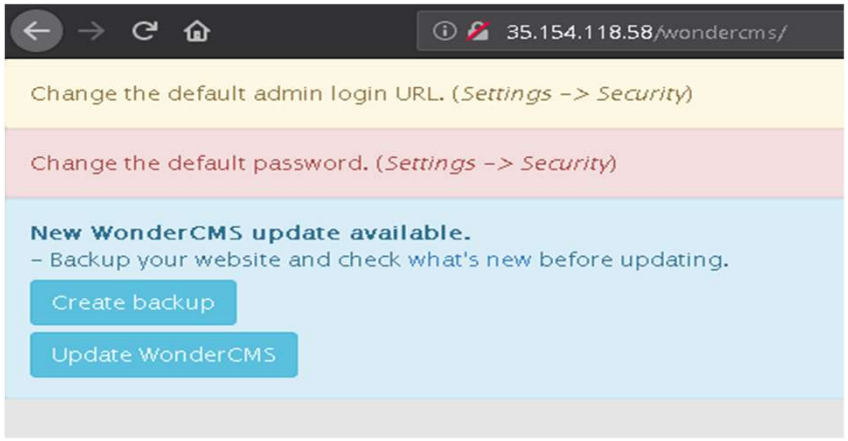
14. Components with Known Vulnerabilities:

Outdated Elements (Moderate)	**Affected Elements :** • PHP • WonderCMS

Observations:

The PHP version installed is not the latest one and has multiple vulnerabilities that can be exploited. Also, WonderCMS is also outdated and highly vulnerable.





Business Impact(High):

Exploits of every vulnerability detected is regularly made public and hence outdated software can very easily be taken advantage of. If the attacker comes to know about this vulnerability, he may directly use the exploit to take down the entire system, which is a big risk.

Recommendation:

1. Upgrade to the latest version of Affected Software/theme/plugin/OS which means latest version number
2. If upgrade is not possible for the time being, isolate the server from any other critical data and servers

References:

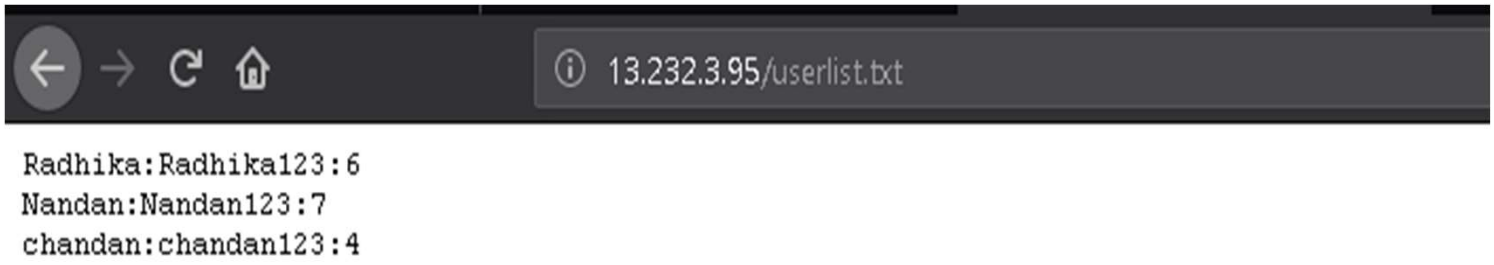
- https://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/version_id-298515/PHP-PHP-5.6.39.html
- https://www.owasp.org/index.php/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities

15. Forced Browsing Flaw

Default Debug Pages (Moderate)	Below mentioned URLs have improper server side filter. The Vulnerability is moderate because there is a password leakage vulnerability. **Affected URL :** • http://35.154.118.58/ Default pages available: • robots.txt • sever-status • phpinfo.php • userlist.txt • ovidentiaCMS

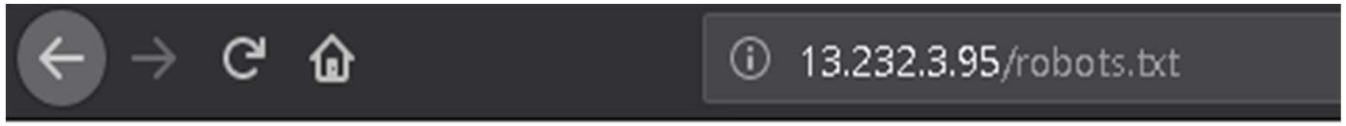
Proof Of Concept(PoC):

When we typed and entered userlist.txt at the end:



Proof Of Concept(PoC):

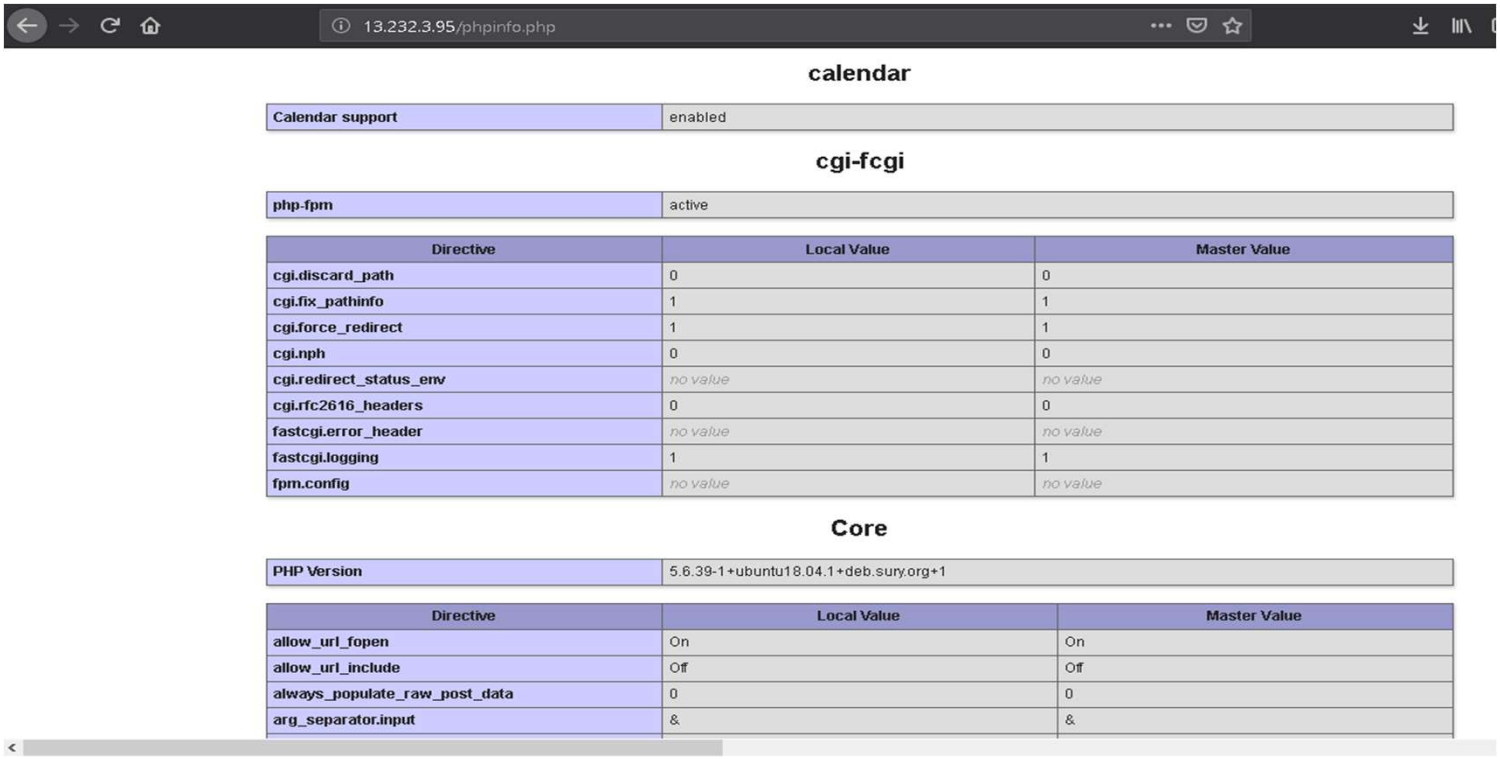
When we entered robots.txt at the end of the index page URL, we got:




```
User-Agent: *
Disallow: /static/images/
Disallow: /ovidentiaCMS
```

Proof Of Concept(PoC):

When we entered phpinfo.php at the end ,we got:



calendar

Calendar support	enabled

cgi-fcgi

php-fpm	active

Directive	Local Value	Master Value
cgi.discard_path	0	0
cgi.fix_pathinfo	1	1
cgi.force_redirect	1	1
cgi.nph	0	0
cgi.redirect_status_env	*no value*	*no value*
cgi.rfc2616_headers	0	0
fastcgi.error_header	*no value*	*no value*
fastcgi.logging	1	1
fpm.config	*no value*	*no value*

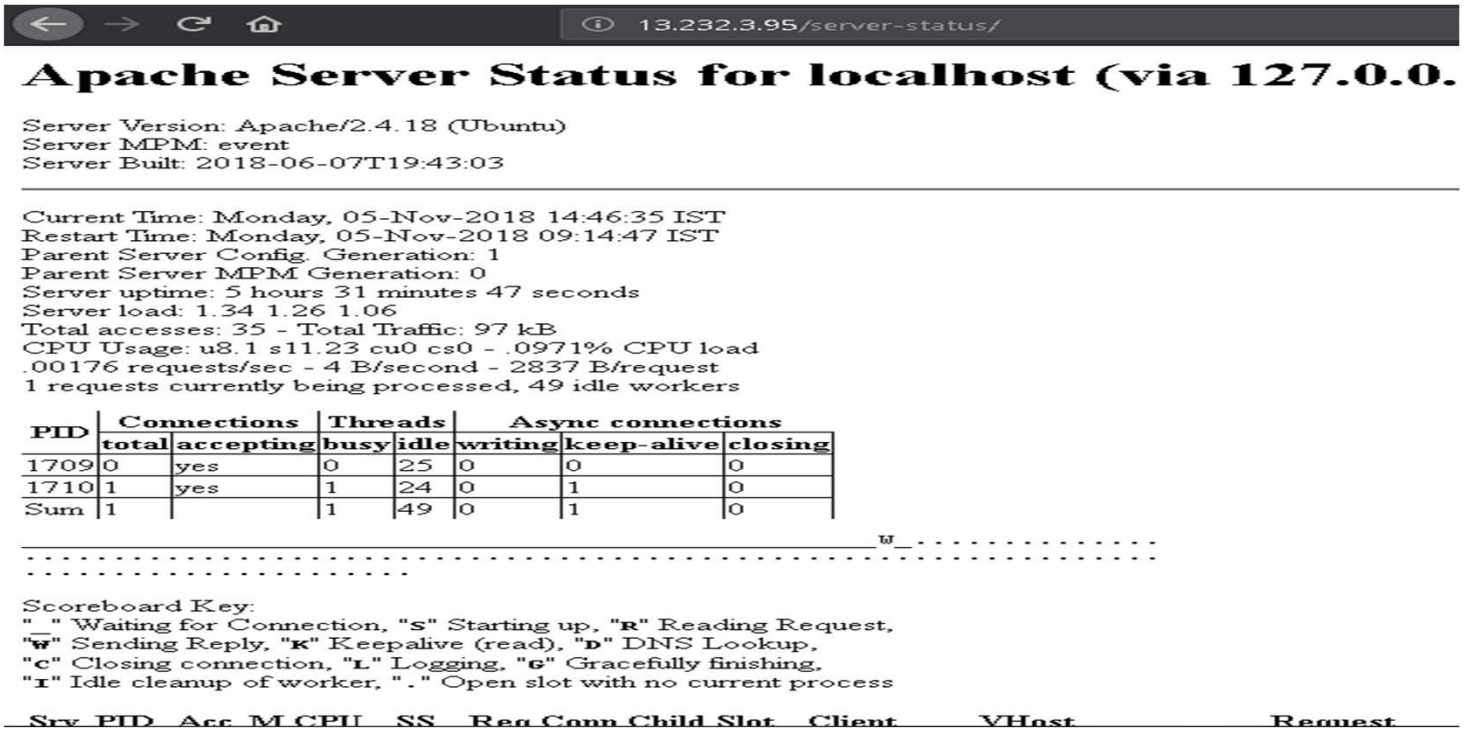
Core

PHP Version	5.6.39-1+ubuntu18.04.1+deb.sury.org+1

Directive	Local Value	Master Value
allow_url_fopen	On	On
allow_url_include	Off	Off
always_populate_raw_post_data	0	0
arg_separator.input	&	&

Proof Of Concept(PoC):

When we entered server-info at the end of index URL:



Business Impact(Low):

Although this vulnerability does not have a direct impact to users or the server, though it can help the attacker in mapping the server architecture and plan further attacks on the server

Recommendation:

Take the following precautions:

- Disable all default pages and folders including server-status and server-info

References:

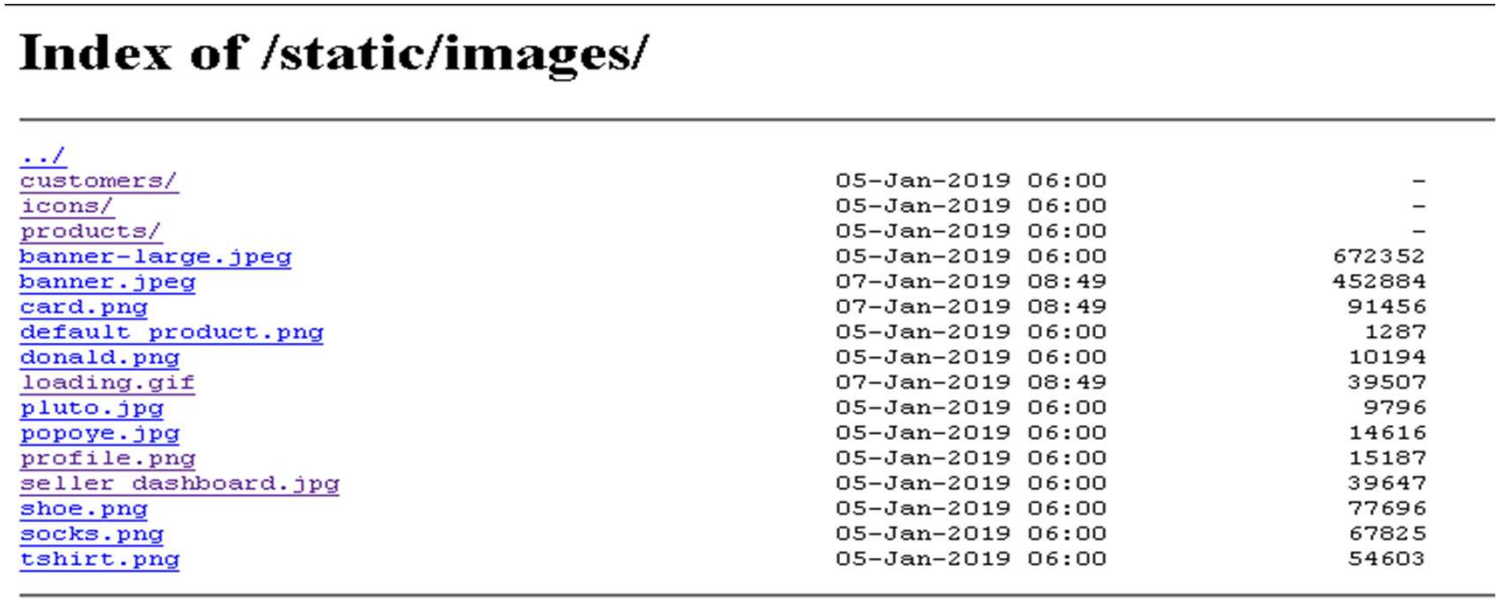
- *https://vuldb.com/?id.88482*
- *https://httpd.apache.org/docs/current/mod/mod_status.html*
- *https://www.beyondsecurity.com/scan_pentest_network_vulnerabilities_apache_http_server_httponly_cookie_information_disclosure*

16. Directory Listing

Directory Listing (Low)	Below mentioned parameters are vulnerable to Directory Listing **Affected URL :** • http://35.154.118.58/static/images/

Observation:

1. Navigate to http://35.154.118.58/static/images/
2. Complete listing of directory is shown containing names of files and folders of the website.



Index of /static/images/

../		
customers/	05-Jan-2019 06:00	-
icons/	05-Jan-2019 06:00	-
products/	05-Jan-2019 06:00	-
banner-large.jpeg	05-Jan-2019 06:00	672352
banner.jpeg	07-Jan-2019 08:49	452884
card.png	07-Jan-2019 08:49	91456
default product.png	05-Jan-2019 06:00	1287
donald.png	05-Jan-2019 06:00	10194
loading.gif	07-Jan-2019 08:49	39507
pluto.jpg	05-Jan-2019 06:00	9796
popoye.jpg	05-Jan-2019 06:00	14616
profile.png	05-Jan-2019 06:00	15187
seller dashboard.jpg	05-Jan-2019 06:00	39647
shoe.png	05-Jan-2019 06:00	77696
socks.png	05-Jan-2019 06:00	67825
tshirt.png	05-Jan-2019 06:00	54603

Observation:

- Navigate to http://35.154.118.58/static/images/icons/
- This gives the complete list of icons.

Business Impact – Moderate

- Although this vulnerability does not have a direct impact to users or the server, though it can ai d the attacker with information about the server and the users

- Also, attacker can simply download the backups and images and view them.

Recommendation:

Take the following precautions:

- Disable Directory Listing
- Put an index.html in all folders with default message

References:

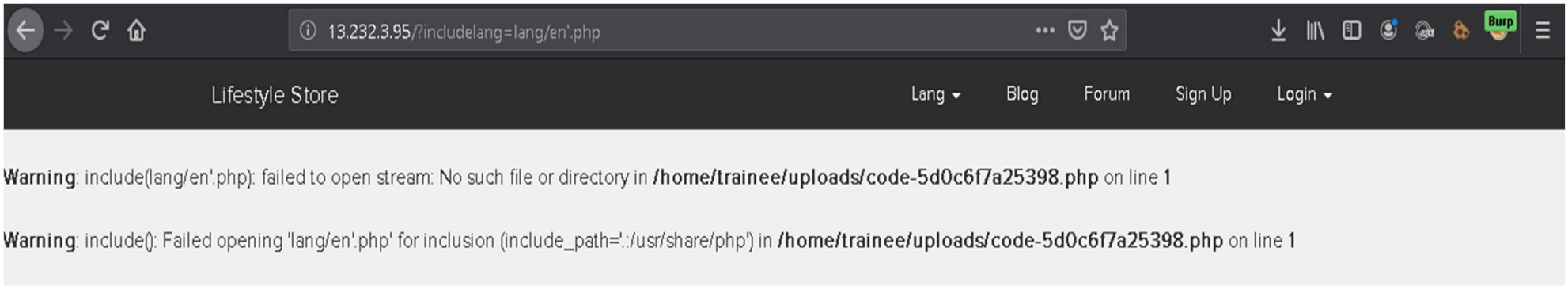
- *https://cwe.mitre.org/data/definitions/548.html*
- https://www.netsparker.com/blog/web-security/disable-directory-listing-web-servers/

17. Default Error Display:

Default Error Display (Low)	Below mentioned urls have default error displaying on fuzzing: **Affected URL :** • http://35.154.118.58/?includelang=lang/en.php **Payload** • en'.php
	Affected URL: • http://35.154.118.58/search/search.php Parameter: • q(GET) Payload • q='

Observation:

The default error with the path is displayed as:



Proof Of Concept(PoC):

When we give ' in the search option of the home page, we get the error as:

Business Impact(Moderate):

Although this vulnerability does not have a direct impact to users or the server, though it can help the attacker in mapping the server architecture and plan further attacks on the server.

Recommendation:

Do not display the default error messages because it not tells about the server but also sometimes about the location. So, Whenever there is an error ,send it to the same page or throw some manually written error.

References:

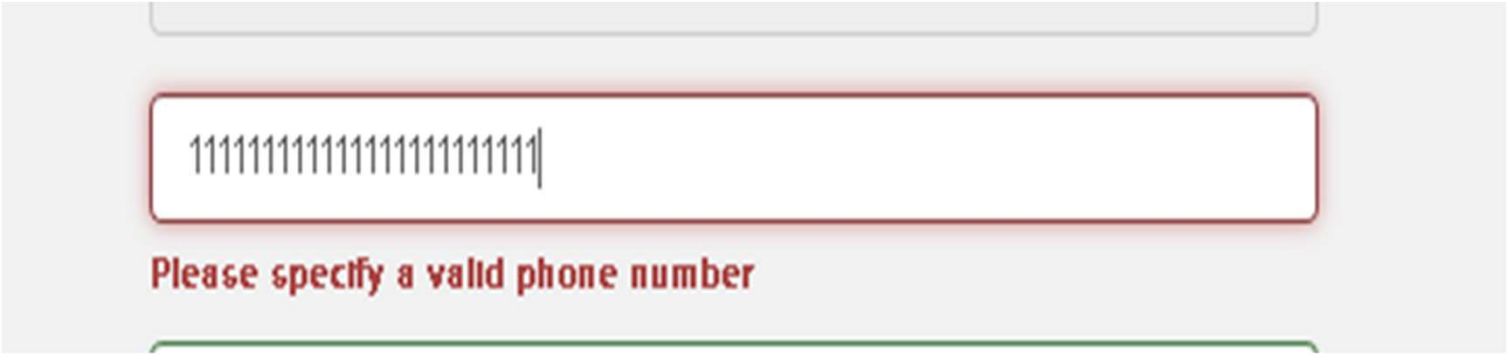
https://www.owasp.org/index.php/Improper_Error_Handling

18. Client Side Filter Bypass:

Improper Server Side Filter (Low)	Below mentioned URLs have improper server side filter **Affected URL :** • http://13.233.108.21/profile/16/edit/ **Affected parameter:** • Contact Number

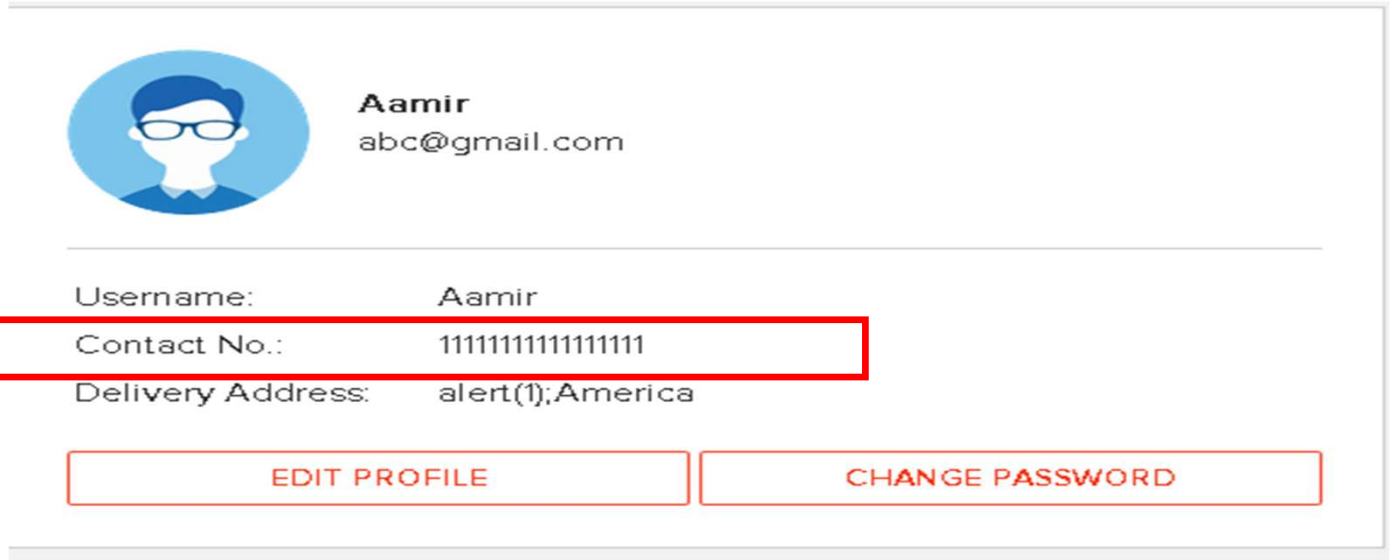
Observation:

After logging in as customer, when we try to edit our phone number to some invalid one, the error is as shown.



Proof Of Concept(PoC):

But when we give a valid phone number on the client side,but intercept it through burpsuite and again give invalid number,it gets accepted.



Business Impact(Low):

The data provided by the user must be checked for proper validator information as it could result in saving false data of the customer.

Recommendation:

1. Implement all critical checks on server side code only
2. Client-side checks must be treated as decorative only
3. All business logic must be implemented and checked on the server code. This includes user input, the flow of applications and even the URL/Modules a user is supposed to access or not

References:

http://projects.webappsec.org/w/page/13246933/Improper%20Input%20Handling

https://www.owasp.org/index.php/Unvalidated_Input

Thank You

For any further clarifications/patch assistance, please contact:
6296440247