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	Moderate	Directory Listing	2
13	Moderate	PII Leakage	1
14	Severe	Open Redirection	1
15	Severe	Bruteforce Exploitation of Coupon Codes	1
16	Critical	Command Execution Vulnerability	2
17	Severe	Forced Browsing	2
18	Severe	Cross-Site Request Forgery	2
19	Critical	Seller Account Access	1

1.SQL Injection

Below mentioned URL in the online e-commerce portal is vulnerable to SQL injection attack

Affected URL :

- http://13.234.115.86/products.php?cat=1

Affected Parameters :

- cat (GET parameter)

Payload:

- cat=1’

Other similar SQLi in the application

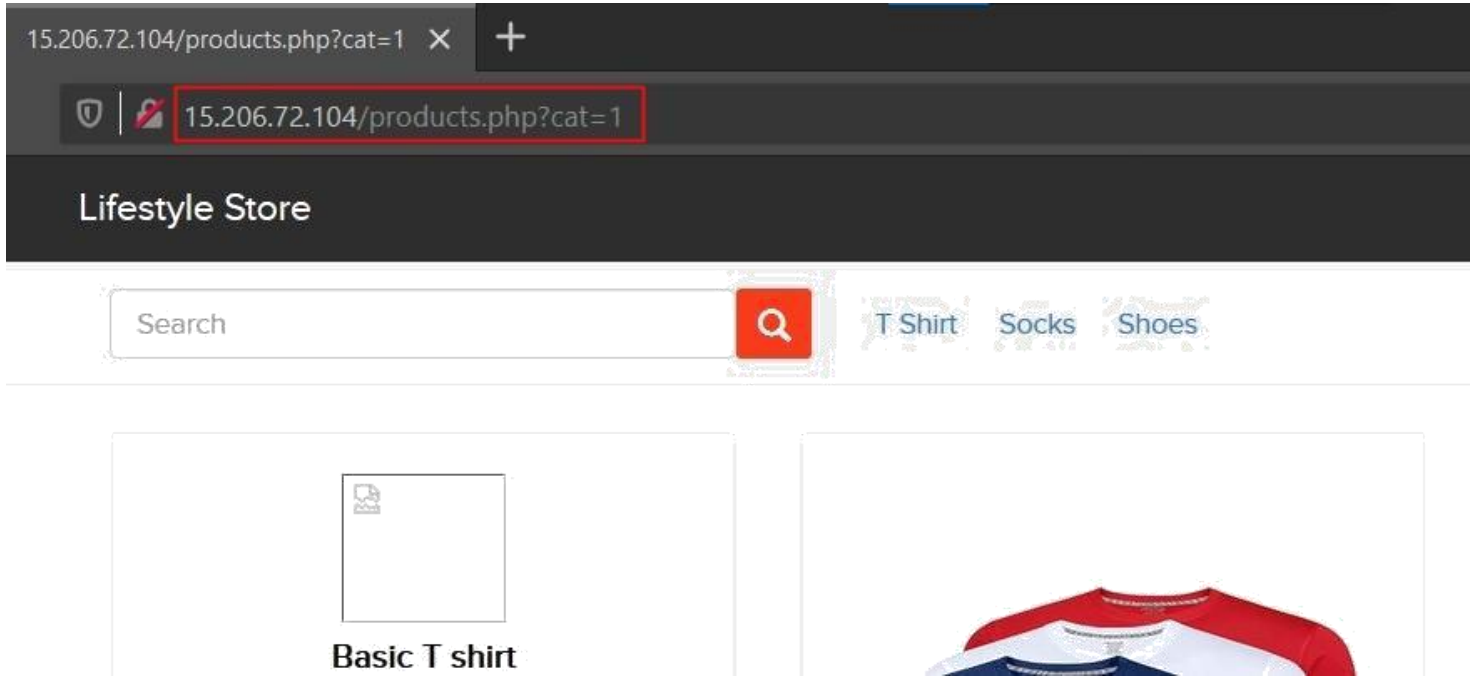
Affected URL :

- http://13.234.115.86/products.php?cat=2

http://13.234.115.86/products.php?cat=3

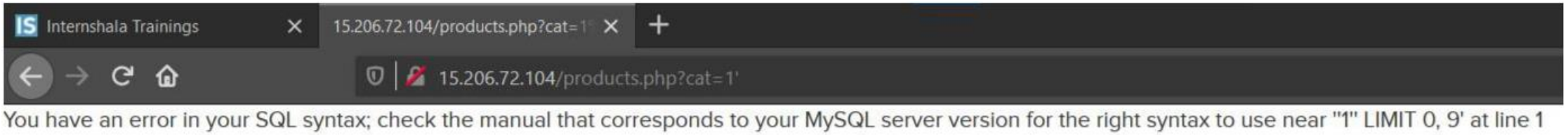
Observation

- Navigate 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.



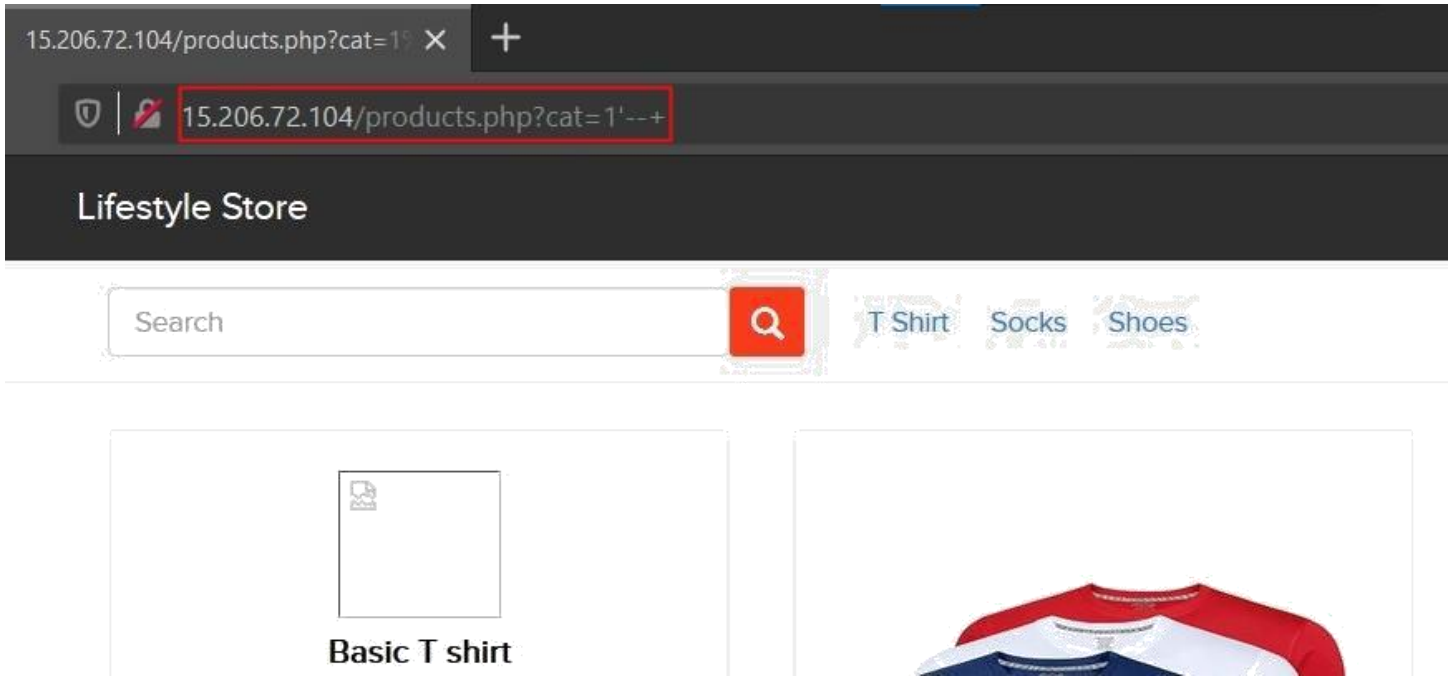
Observation

- 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.



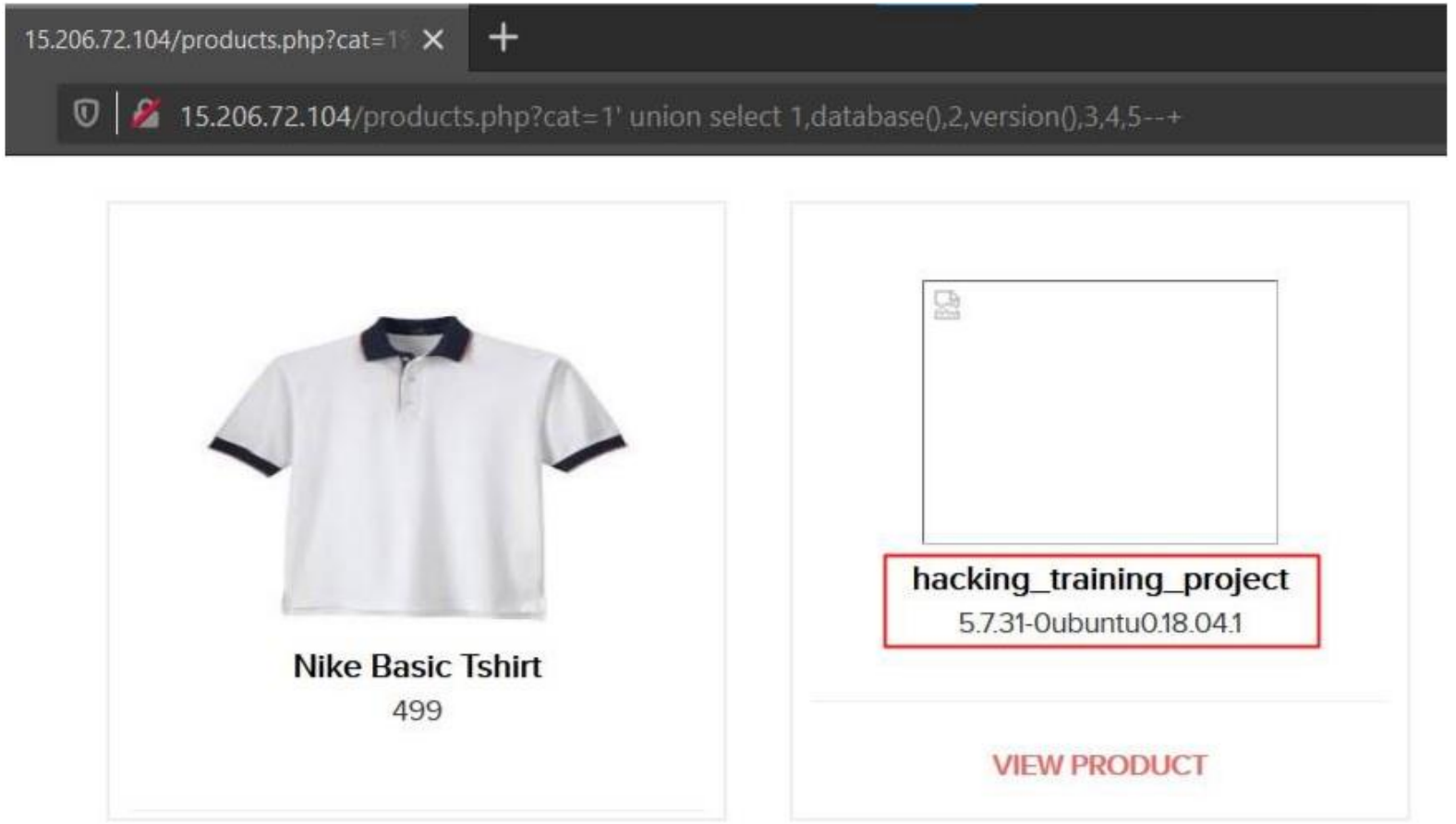
Observation

- We then put --+ : 15.206.72.104/products.php?cat=1’--+ and the error is removed confirming SQL injection:



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://15.206.72.104/products.php?cat=1’ union select 1,database(),2,version(),3,4,5 --+



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)

Below mentioned parameters are vulnerable to reflected XSS,

Affected URL :

- http://3.6.40.63/search/search.php?q=(*here*)

Affected Parameters :

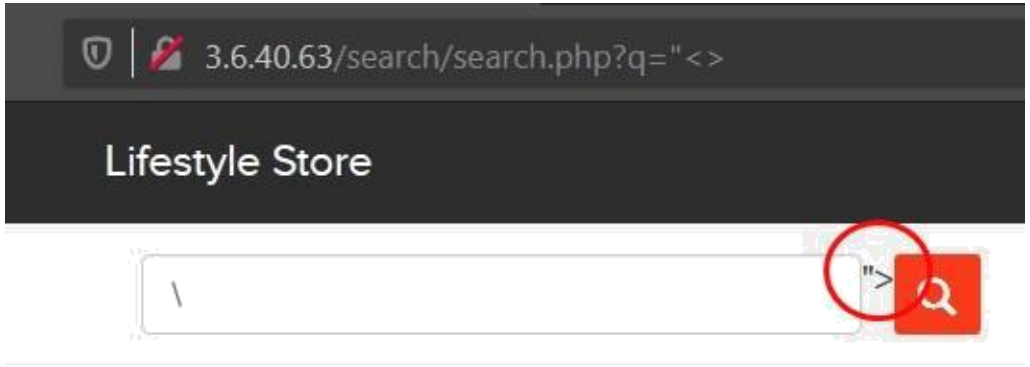
- q

Payload:

- “><script>alert(1)</script>

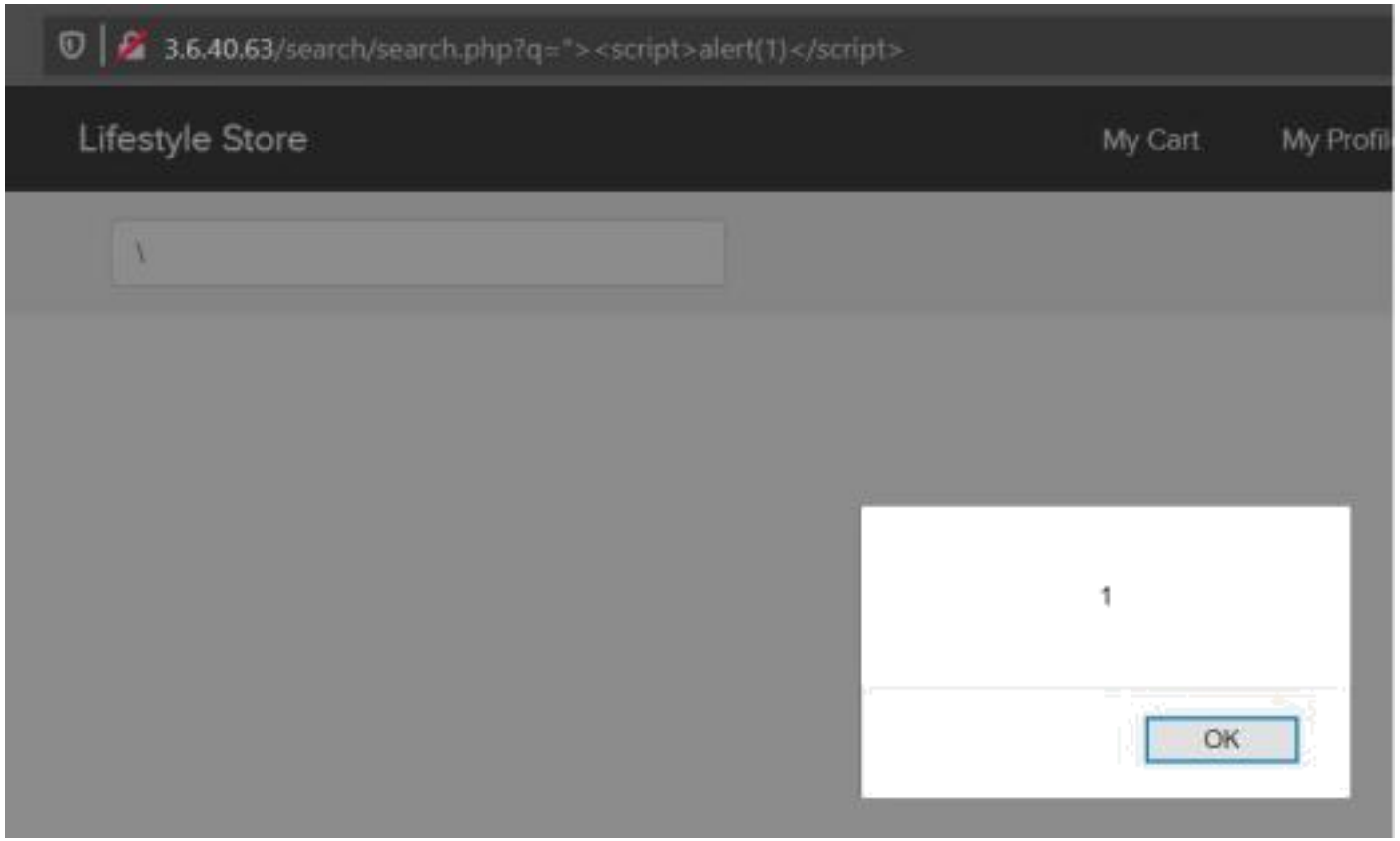
Observation

- 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.



3. Stored Cross Site Scripting (XSS)

Below mentioned parameters are vulnerable to stored XSS,

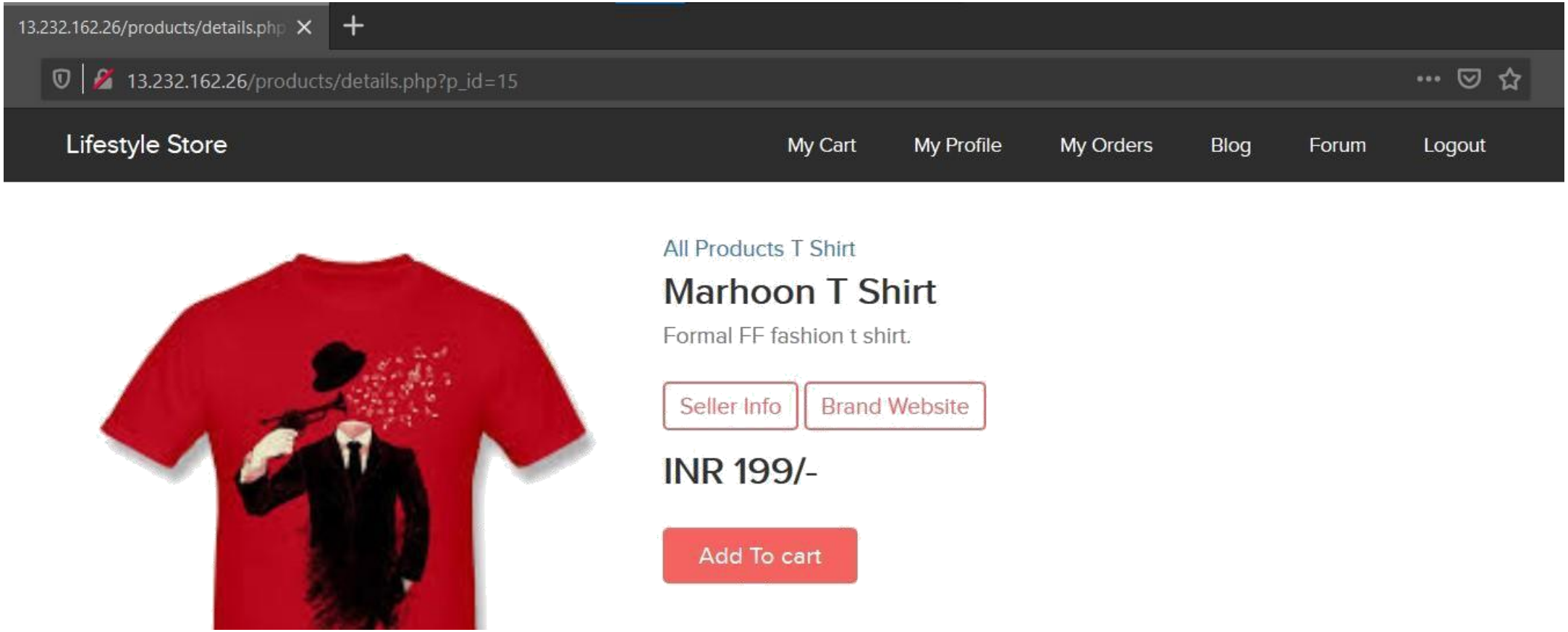
Affected URL :
- http://13.232.162.26/products/details.php?p_id=(*all id's*)

Affected Parameters :
- customer review text field

Payload:
- <script>alert(1)</script>

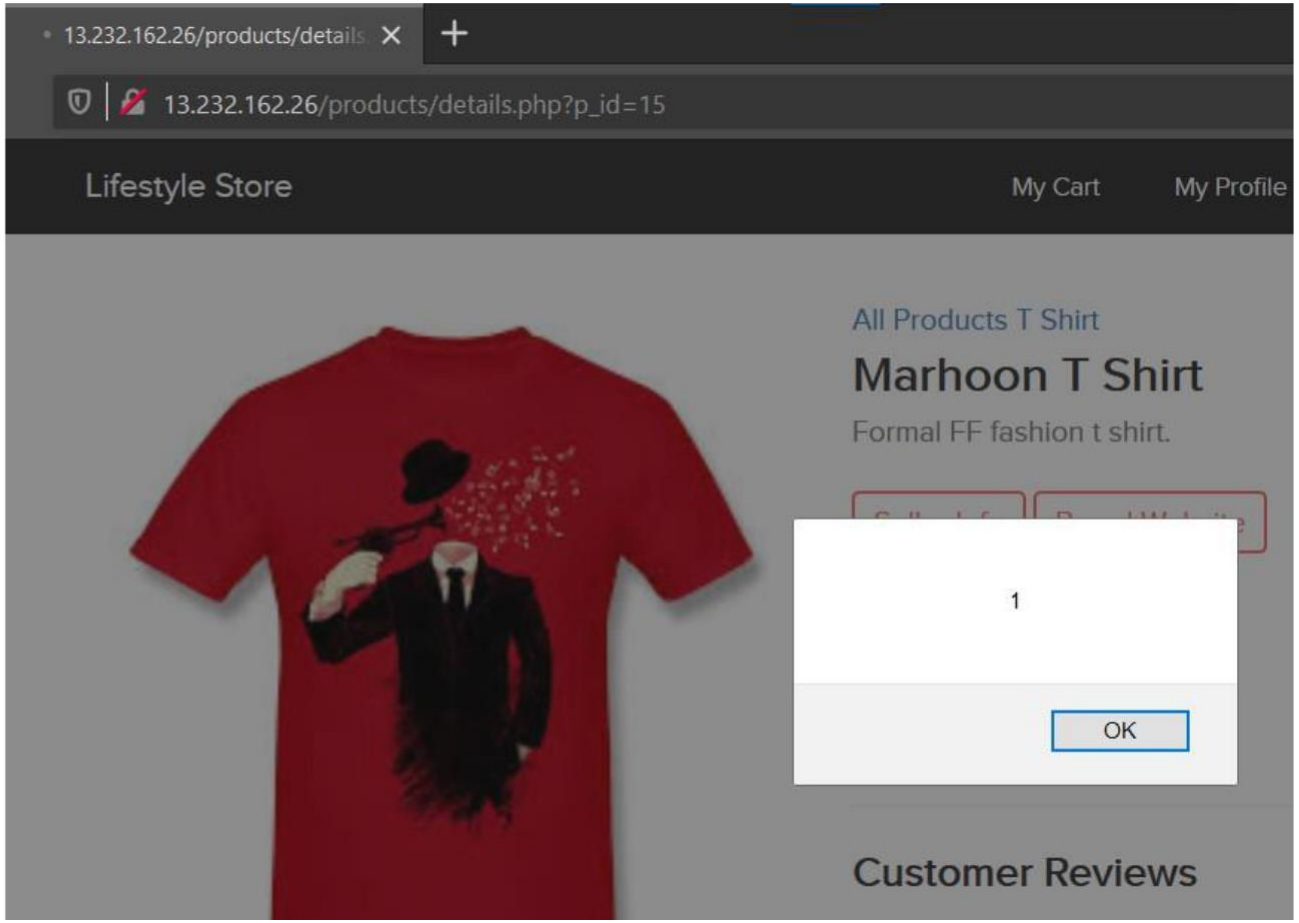
Observation

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://13.232.162.26/products/details.php?p_id=15 (here I selected product number 15).

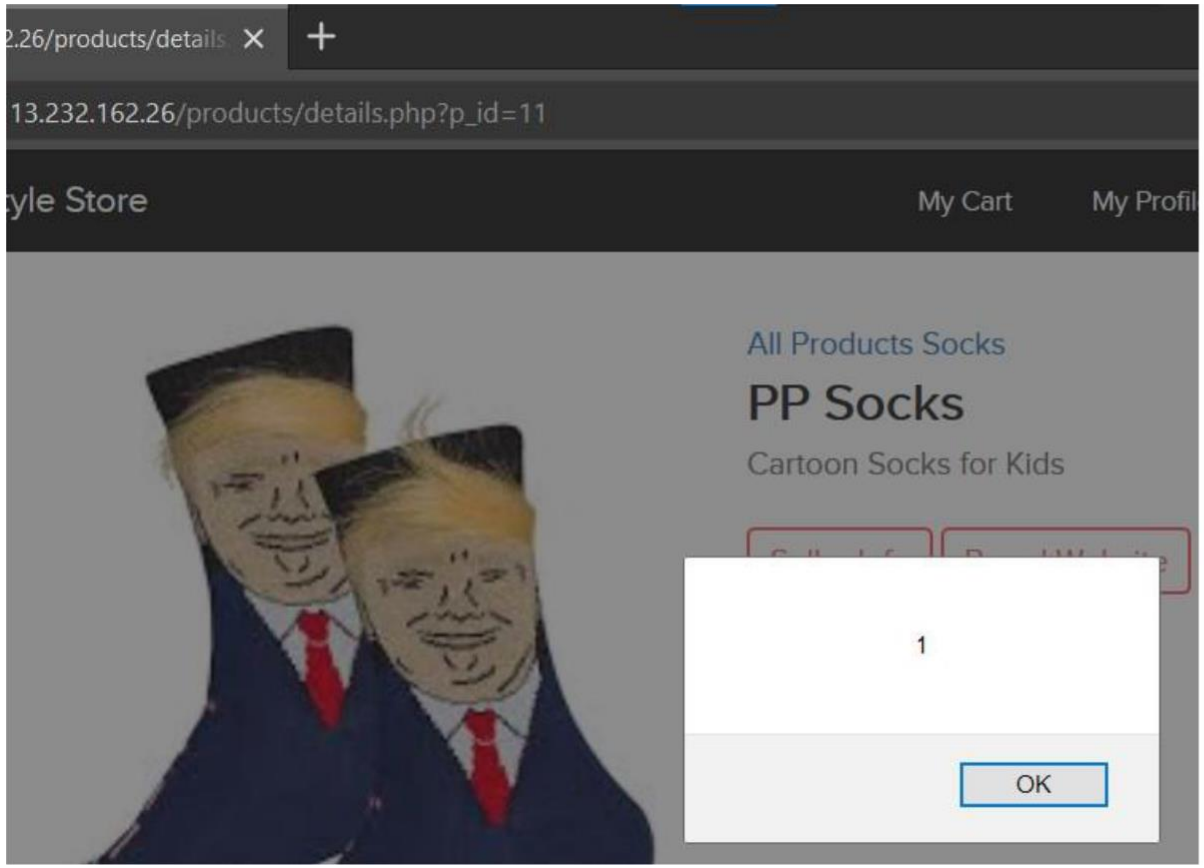


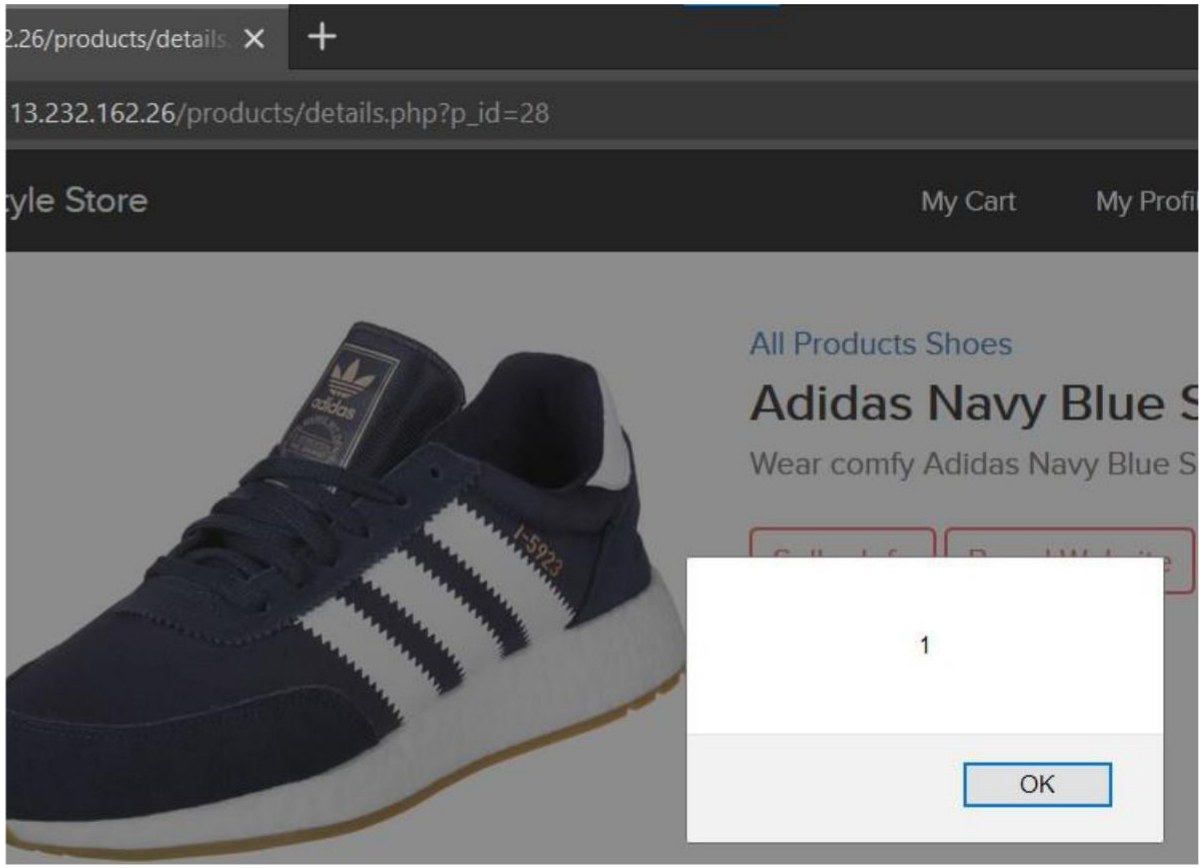
PoC – the script was executed

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.



PoC


2.26/products/details
13.232.162.26/products/details.php?p_id=11
yle Store
My Cart
My Profil
All Products Socks
PP Socks
Cartoon Socks for Kids
1
OK


2.26/products/details
13.232.162.26/products/details.php?p_id=28
yle Store
My Cart
My Profi
All Products Shoes
Adidas Navy Blue S
Wear comfy Adidas Navy Blue S
1
OK

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

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

Affected URL :

- http://13.127.165.218/orders/orders.php?customer=(*all customer id's*)

Affected Parameters :

- customer (GET parameters)

Affected URL :

- http://13.127.165.218/products/details.php?p_id=(*all id's*)
- http://3.6.40.63/forum/index.php?u=/user/profile/(*any id*)

Affected Parameters :

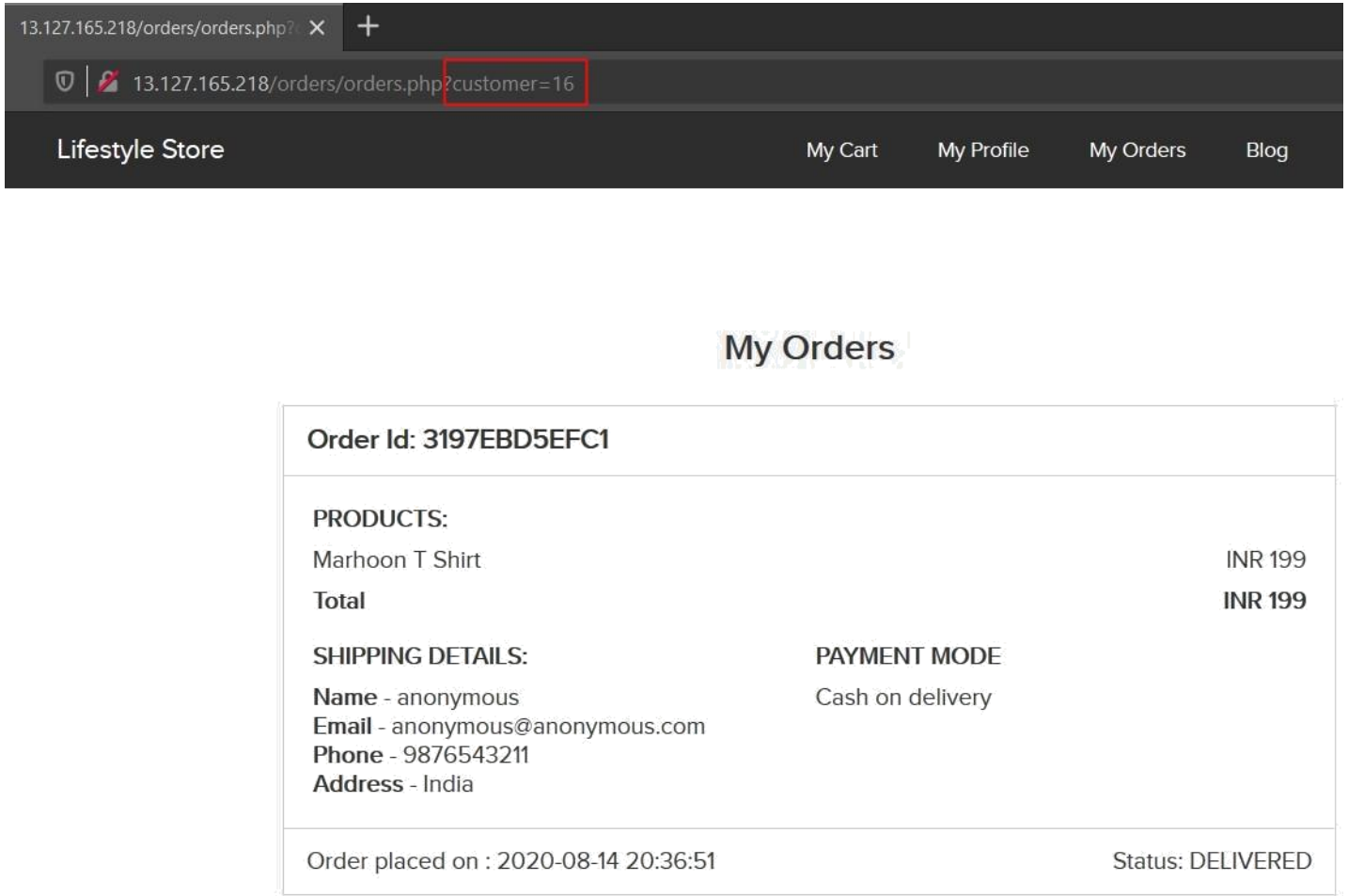
- p_id (GET parameters)
- u=/user/profile/(*any id*

Observation

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

- Notice the URL : http://13.127.165.218/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.



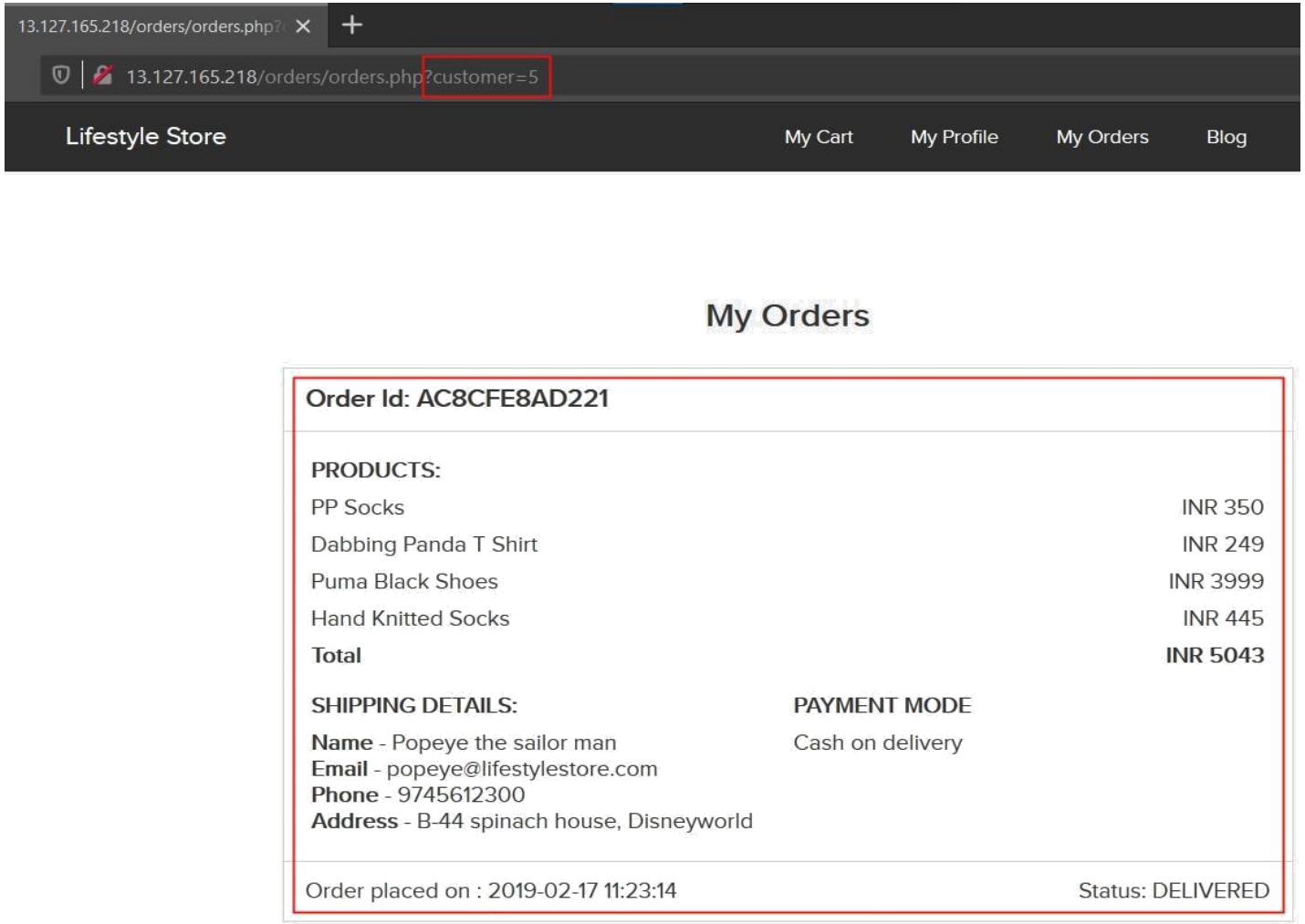
Observation

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

Request	Payload	Status	Error	Timeout	Length	Comment
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	

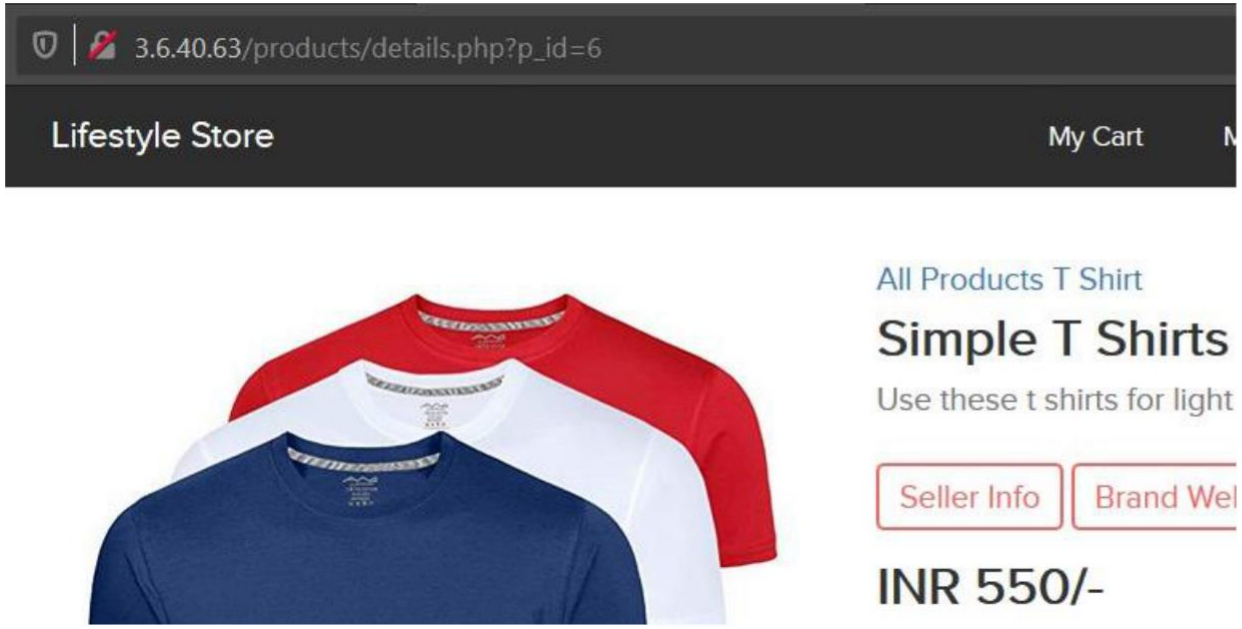
PoC – accessing other customer's details

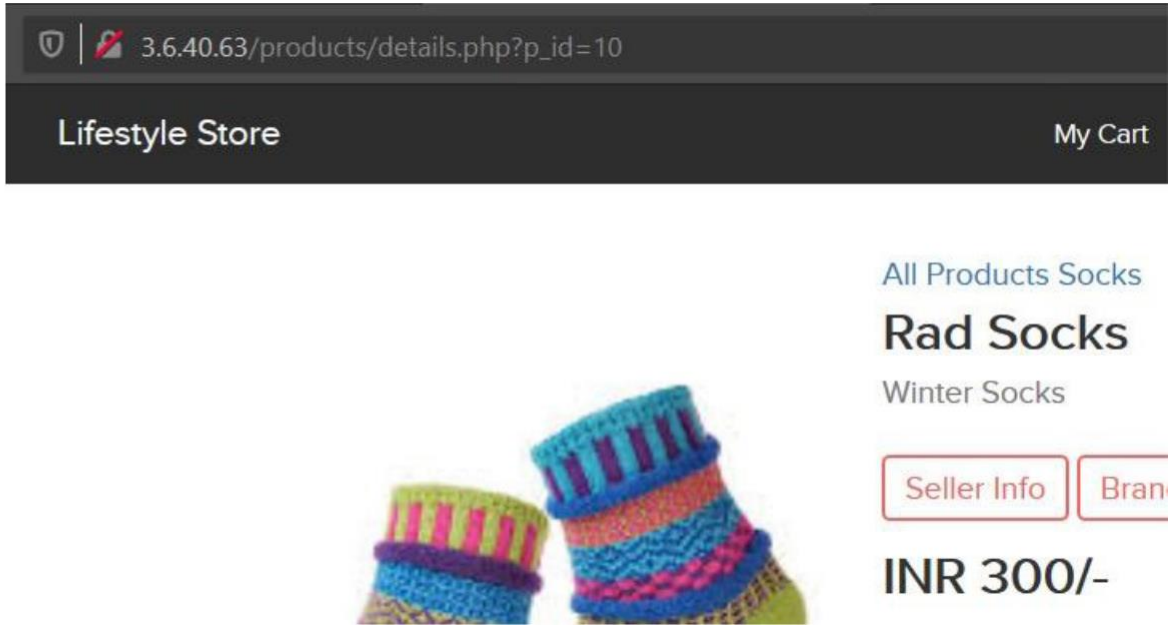
- Now, we change the customer id to 5.
- We get the order details along with shipping details and payment mode of other customers(here the user with customer id = 5).



PoC

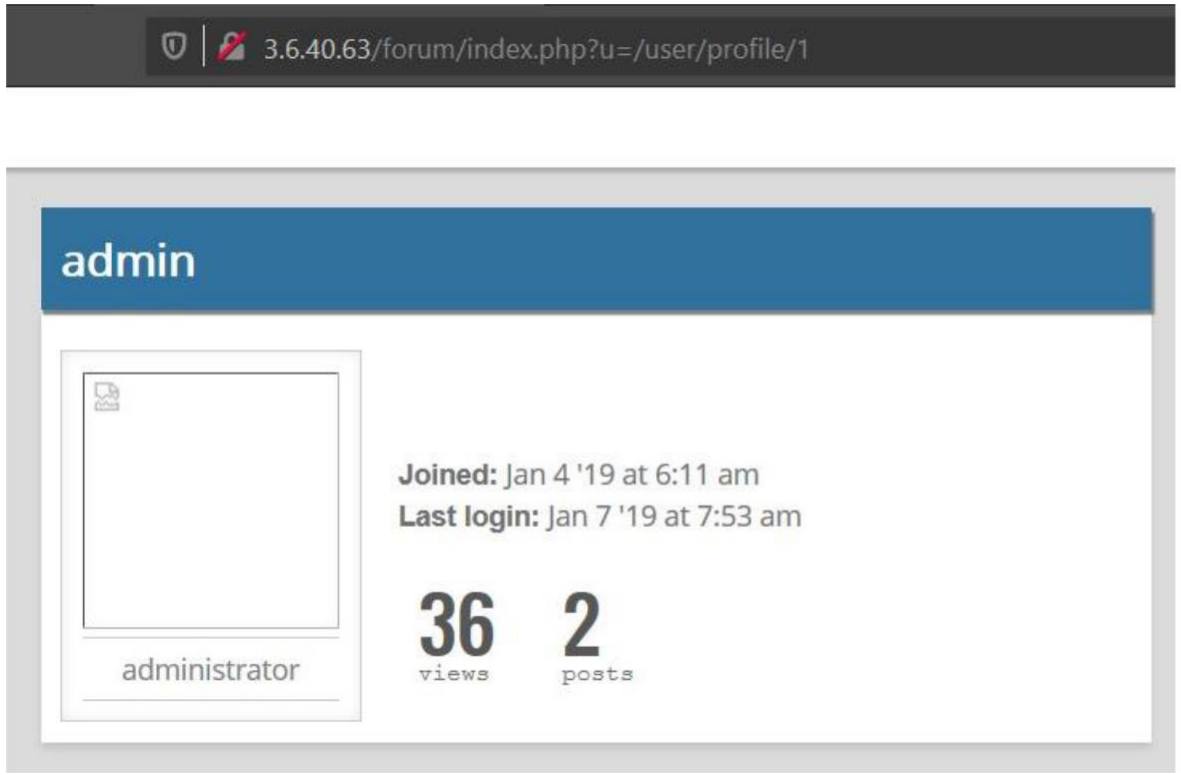
- Just by changing the *product id*, other products can be seen.

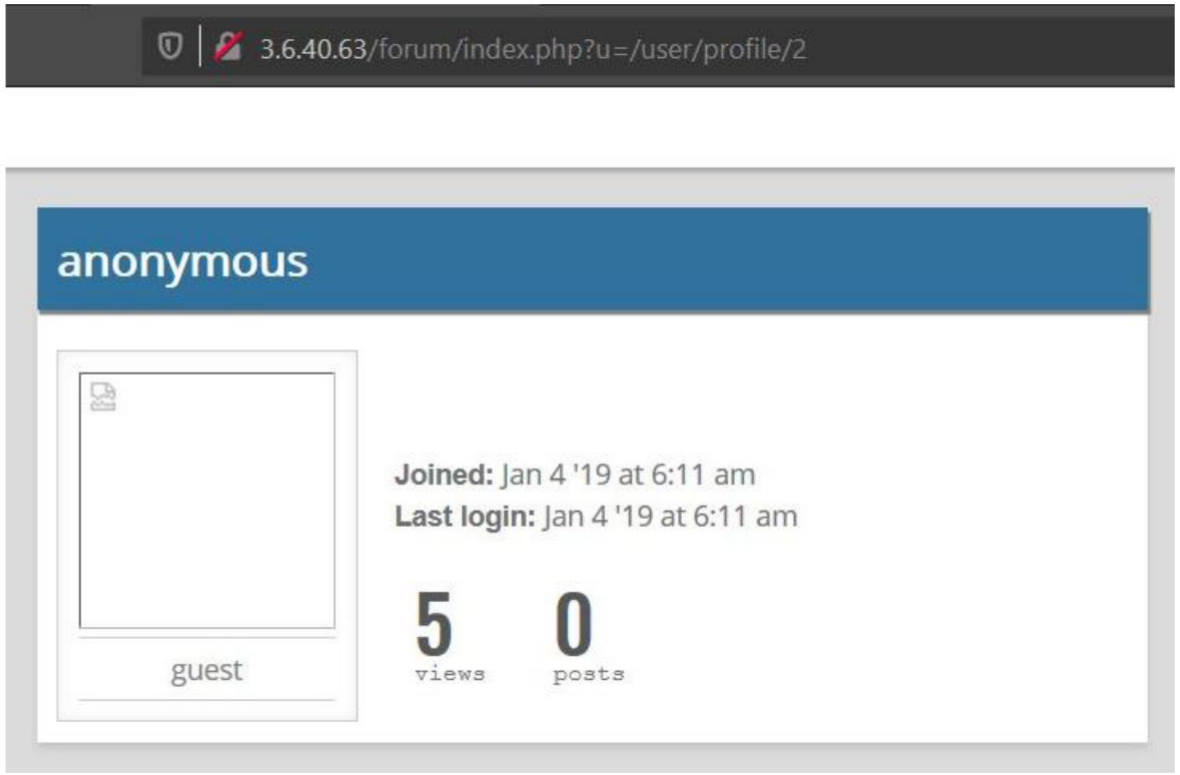




PoC

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





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 user 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

The below mentioned login page allows login via OTP which can be brute forced

Affected URL :

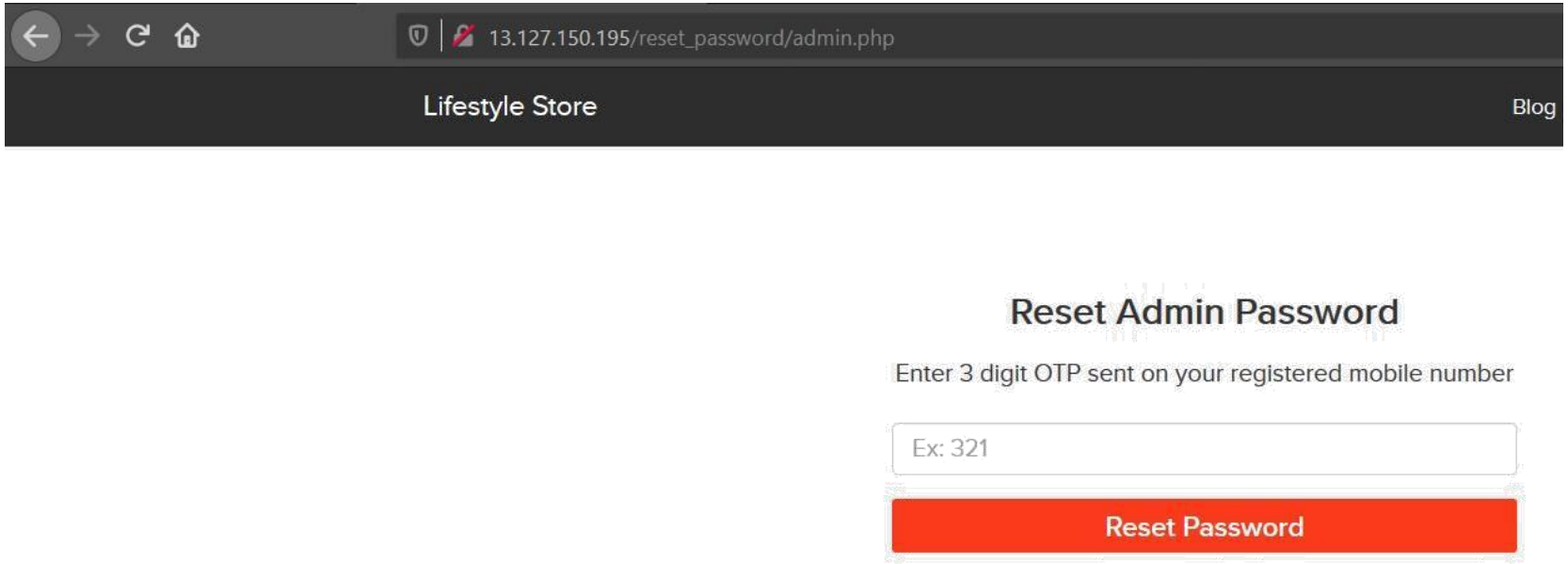
- http://13.127.150.195/login/admin.php

Affected Parameters :

- otp (POST parameters)

Observation

- Navigate to http://13.127.150.195/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).

```
GET /reset_password/admin.php?otp=321 HTTP/1.1
Host: 13.127.165.218
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
DNT: 1
Connection: close
Referer: http://13.127.165.218/reset_password/admin.php
Cookie: key=552ABD04-CFD0-C7D1-748F-BC95609DB4BA; PHPSESSID=f0oo867v5u2bl19sdmr3f104f1; X-XSRF-TOKEN=
970697408eea306a899b13e749a74c0593229c81e44484d97ffcfb9d2078e1cf
Upgrade-Insecure-Requests: 1
```

Observation

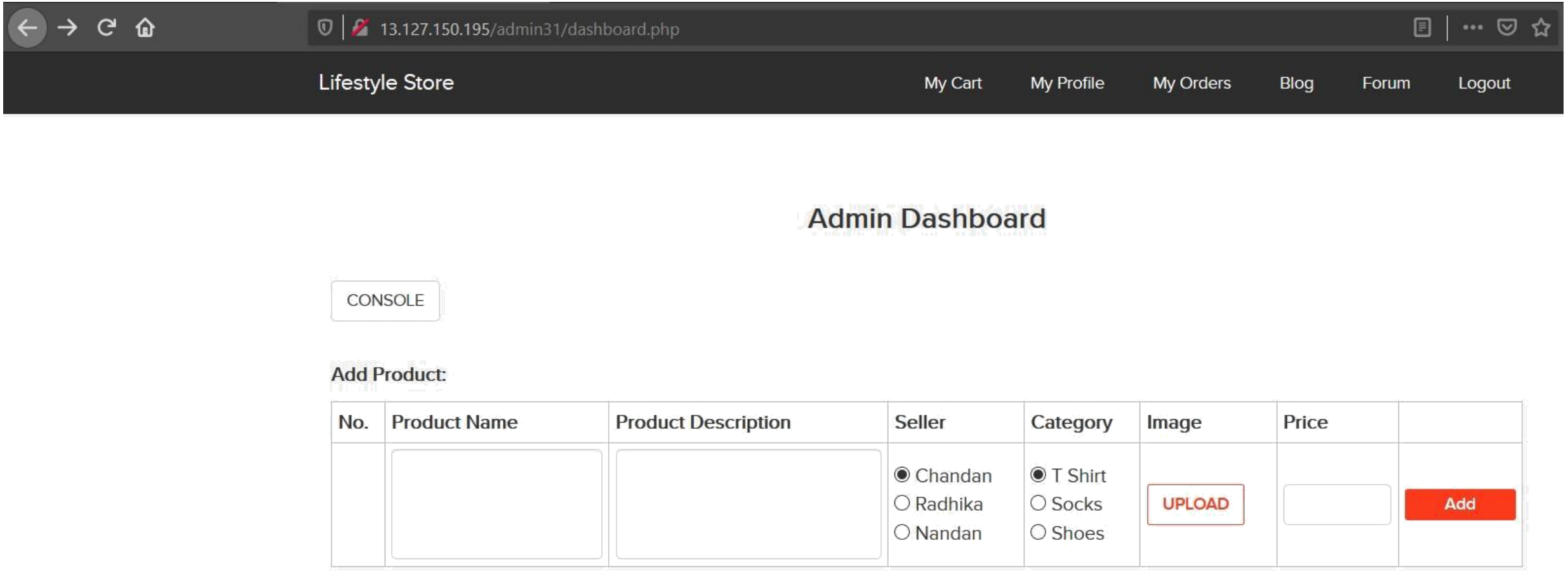
- We shoot the request with all possible combinations of 3 Digit OTPs and upon a successful hit, we get a response containing user details(i.e. the correct OTP). We can use this OTP to reset admin password and then use the new admin password to login as administrator.

- OTP for this Session was 760.

```
1  GET /reset_password/admin.php?otp=§321§ HTTP/1.1
2  Host: 13.127.165.218
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  DNT: 1
8  Connection: close
9  Referer: http://13.127.165.218/reset_password/admin.php
10 Cookie: key=552ABD04-CFD0-C7D1-748F-BC95609DB4BA; PHPSESSID=
   f0oo867v5u2bll9sdmr3fl04fl; X-XSRF-TOKEN=
   970697408eea306a899b13e749a74c0593229c81e44484d97ffcfb9d2078e1cf
11 Upgrade-Insecure-Requests: 1
12
```

Request	Payload	Status	Error	Timeout	Length	Comment
654	753	200	☐	☐	4380	
655	754	200	☐	☐	4380	
656	755	200	☐	☐	4380	
657	756	200	☐	☐	4380	
658	757	200	☐	☐	4380	
659	758	200	☐	☐	4380	
660	759	200	☐	☐	4380	
661	760	200	☐	☐	4476	
662	761	200	☐	☐	4380	
663	762	200	☐	☐	4380	
664	763	200	☐	☐	4380	
665	764	200	☐	☐	4380	
666	765	200	☐	☐	4380	
667	766	200	☐	☐	4380	

PoC – access to admin dashboard



Recommendation

Take the following precautions:

- Use proper rate-limiting checks on the no of OTP checking and Generation requests.
- 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

Below mentioned URL is vulnerable to insecure file uploads,

Affected URL :

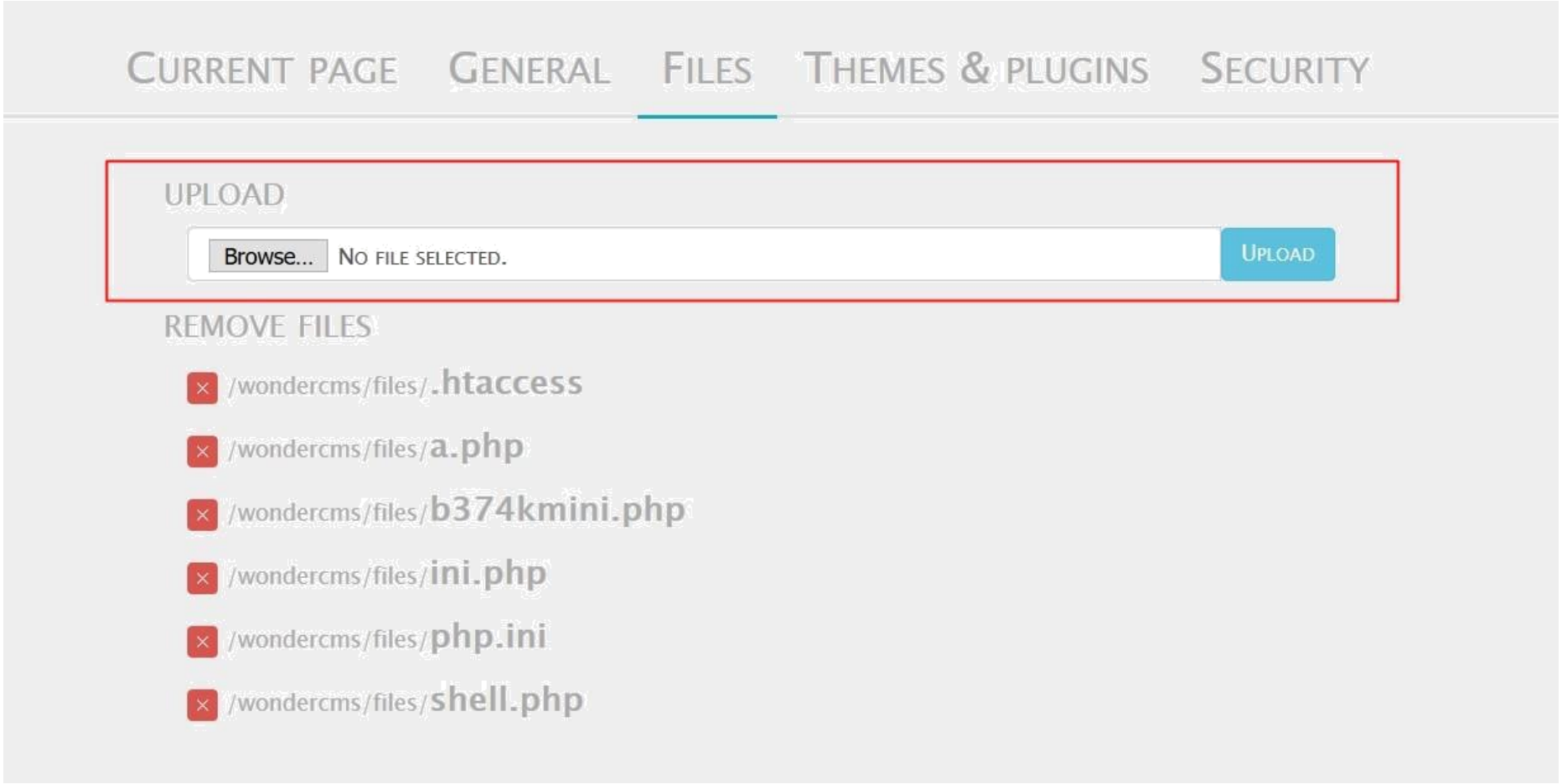
- http://52.66.88.120/wondercms/

File Uploaded :

- backdoor shell (anonymous.php)

Observation

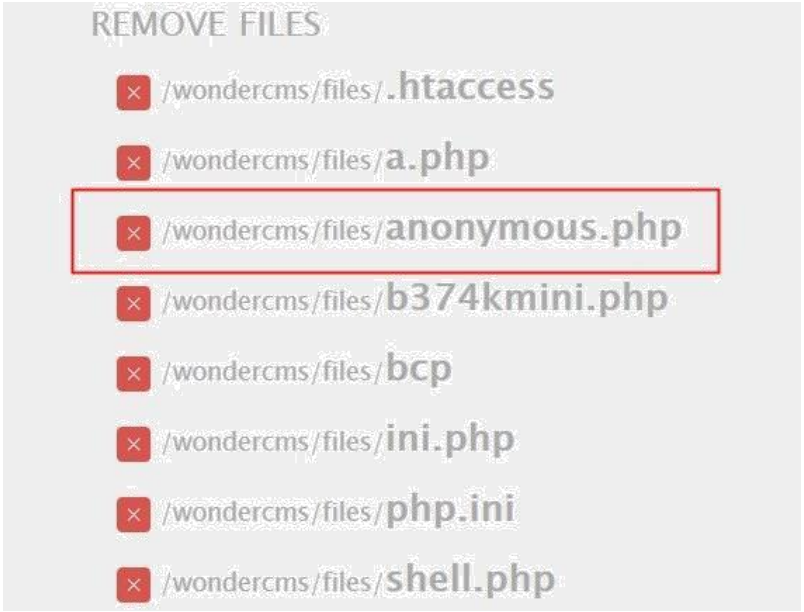
- Navigate to the Blog section of the website and login as admin.
- Now, navigate to the Settings and then go to Files option.
- You will notice an Upload section here,



Observation

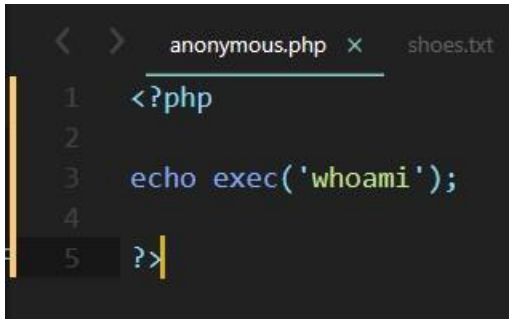
- It looks like we can upload files here, let’s try uploading a file anonymous.php

- And it’s successfully uploaded.

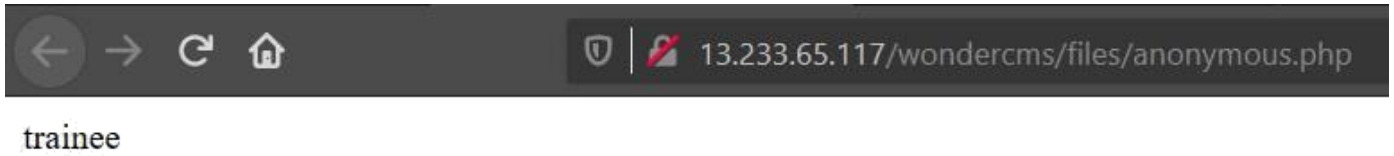


PoC - any command can be executed

- Shell – anonymous.php



- The uploaded shell was executed successfully.



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

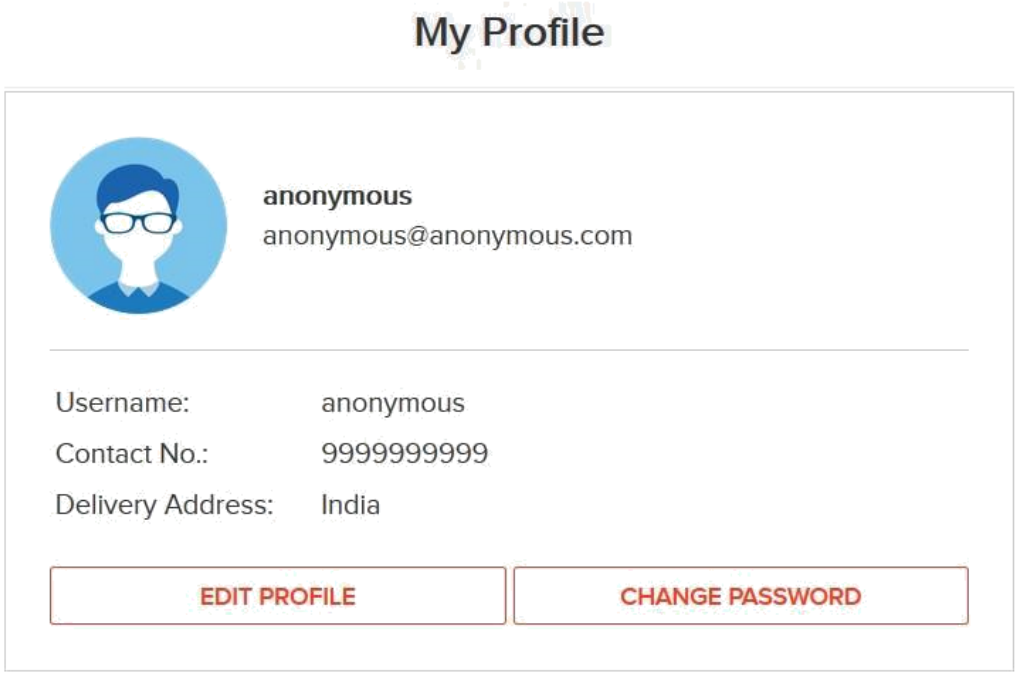
Below mentioned URL is vulnerable to client side filter bypass.

Affected URL :

- http://3.6.40.63/profile/16/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 9876543211 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.

Request

Raw | Params | Headers | Hex

```
POST /profile/submit.php HTTP/1.1
Host: 3.6.40.63
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0
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=---------------------------184845640872487219014071911 23
Content-Length: 714
Origin: http://3.6.40.63
DNT: 1
Connection: close
Referer: http://3.6.40.63/profile/16/edit/
Cookie: key=552ABD04-CFD0-C7D1-748F-BC95609DB4BA; PHPSESSID=947kfipb4g6ijr344mogvtjll4; X-XSRF-TOKEN=
4668653e1659a9972689c2475b72f86478bd20f3ddaf2c7843e0d86f39fa2f60

-----------------------------184845640872487219014071911 23
Content-Disposition: form-data; name="name"

anonymous
-----------------------------184845640872487219014071911 23
Content-Disposition: form-data; name="contact"

1111111111
-----------------------------184845640872487219014071911 23
Content-Disposition: form-data; name="address"

India
-----------------------------184845640872487219014071911 23
Content-Disposition: form-data; name="user_id"

16
-----------------------------184845640872487219014071911 23
Content-Disposition: form-data; name="X-XSRF-TOKEN"

4668653e1659a9972689c2475b72f86478bd20f3ddaf2c7843e0d86f39fa2f60
-----------------------------184845640872487219014071911 23--
```

Response

Raw | Headers | Hex | Render

{"success":true,"successMessage" "Profile updated succesfully."}

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

Below mentioned URL contains components with known vulnerabilities.

Affected URL:

- http://15.206.159.87/wondercms/
- http://15.206.159.87/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

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.														

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

Below mentioned URL is using default admin credentials.

Affected URL:

- http://15.206.159.87/ovidentiaCMS/index.php?tg=login&cmd=authform&msg=Connexion&err=&restricted=1

Component Name:

- ovidentia content management system

Observation

- Navigate to http://15.206.159.87/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:

PoC

- Upon entering the credentials we got the administrator access.

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

Below mentioned URLs shows descriptive error messages,

Affected URL:

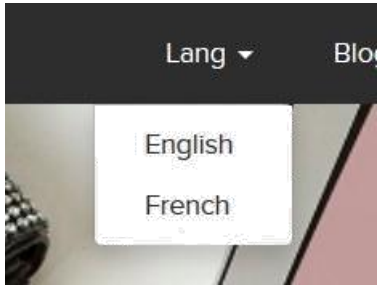
- http://3.6.40.63/?includelang=lang/fr.php

Affected Parameter:

- includelang

Observations

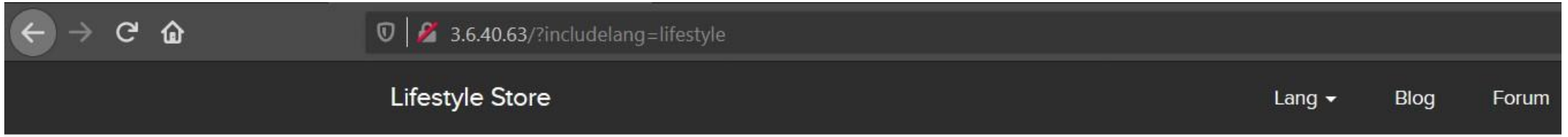
- Navigate to the website and click on change language dropdown, and select any of the two languages.



- 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



Warning: include(lifestyle): failed to open stream: No such file or directory in **/home/trainee/uploads/code-5f3c225757ca4.php** on line **1**

Warning: include(): Failed opening 'lifestyle' for inclusion (include_path='.:/usr/share/php') in **/home/trainee/uploads/code-5f3c225757ca4.php** on line **1**

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

Below mentioned URLs shows default files and pages,

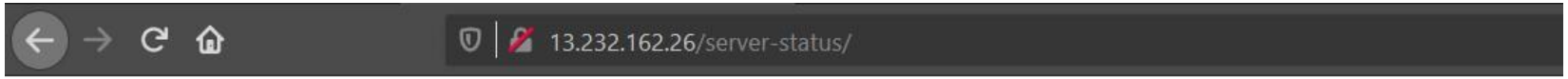
Affected URL:

- http://3.6.40.63/

Default files and pages present:

- server-status
- robots.txt
- userlist.txt
- phpinfo.php

PoC – server-status/



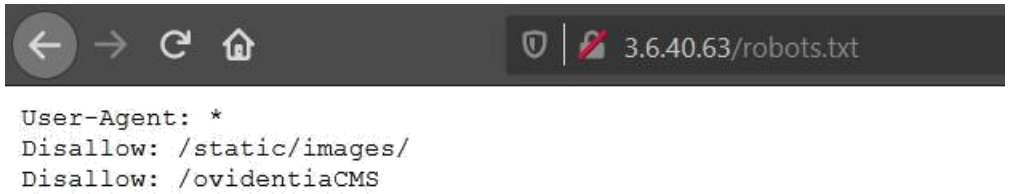
Apache Server Status for localhost (via 127.0.0.1)

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

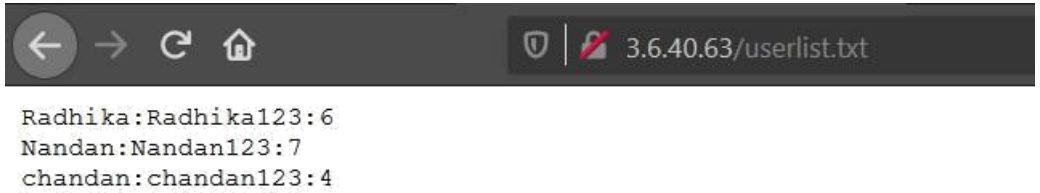
Current Time: Monday, 05-Nov-2018 14:46:35 IST
Restart Time: Monday, 05-Nov-2018 09:14:47 IST
Parent Server Config. Generation: 1
Parent Server MPM Generation: 0
Server uptime: 5 hours 31 minutes 47 seconds
Server load: 1.34 1.26 1.06
Total accesses: 35 - Total Traffic: 97 kB
CPU Usage: u8.1 s11.23 cu0 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

PoC – robots.txt



PoC – userlist.txt



PoC – phpinfo.php

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

System	Linux ip-172-26-6-28 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

Recommendation

Take the following precautions:

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

Referenc
es

11. Remote File Inclusion

Below mentioned URL is vulnerable to RFI.

Affected URL :

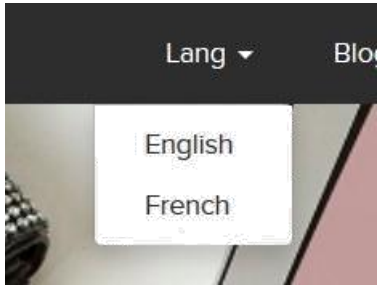
- http://52.66.88.120/?includelang=lang/fr.php

Affected Parameters :

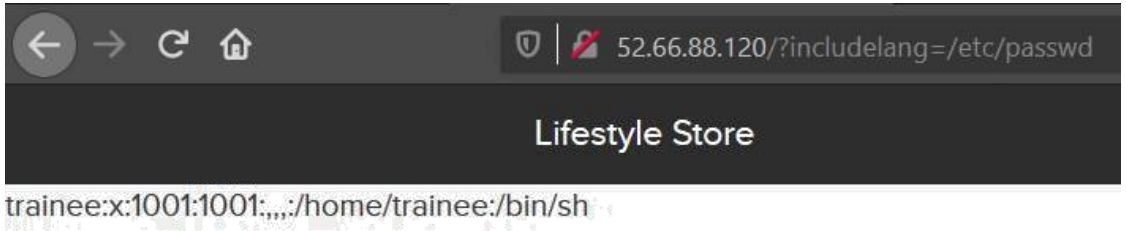
- /etc/passwd (/?includelang=*here*)
- https://www.google.co.in/ (/?includelang=*here*)

Observations

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

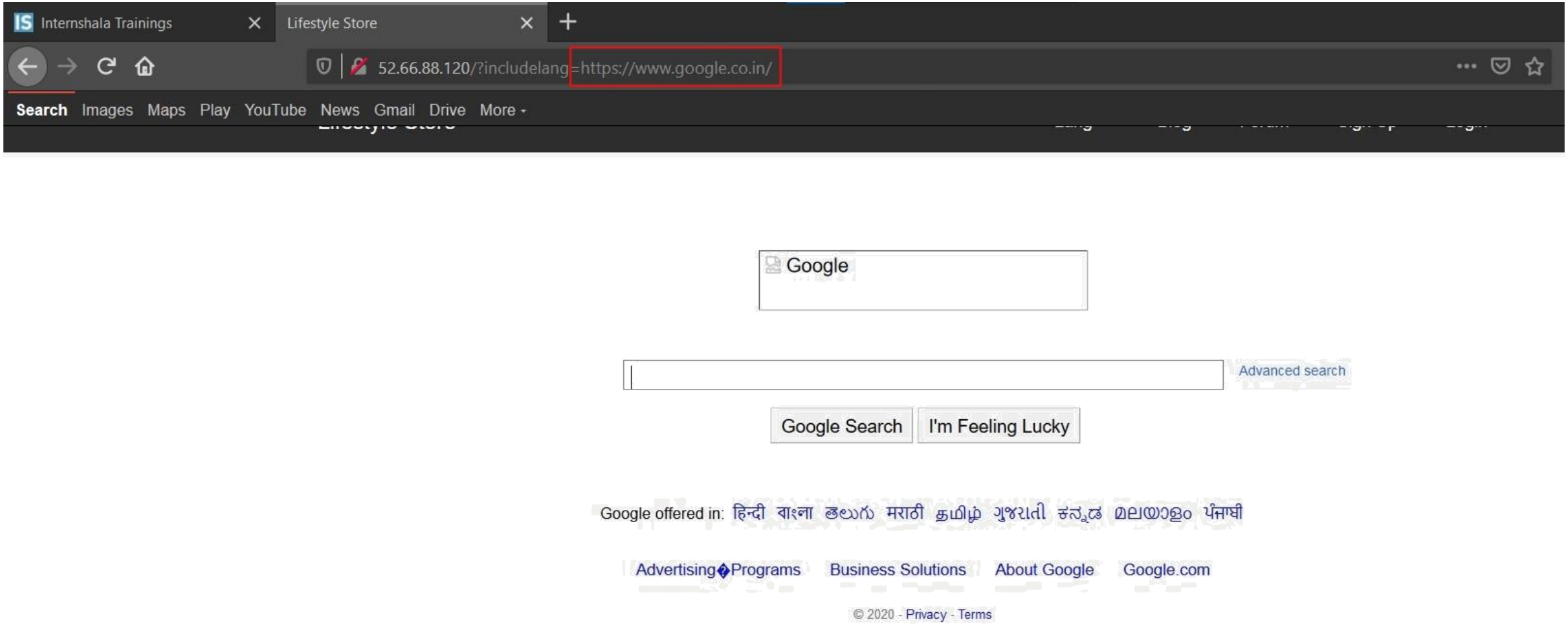


- Now, notice the URL, you get a ‘get’ parameter of includelang which is vulnerable to file inclusion.
- Here, we enter the payload: includelang=/etc/passwd and on executing this file gives us the username.



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.



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

Here are other similar URLs that leaks critical information via directory listing vulnerability.

Affected URL:

- http://13.232.162.26/robots.txt

12. Directory Listing

Below mentioned URL leaks critical information via directory listing vulnerability.

Affected URL:

- http://13.232.162.26/static/images/uploads/products/reebok.jpeg

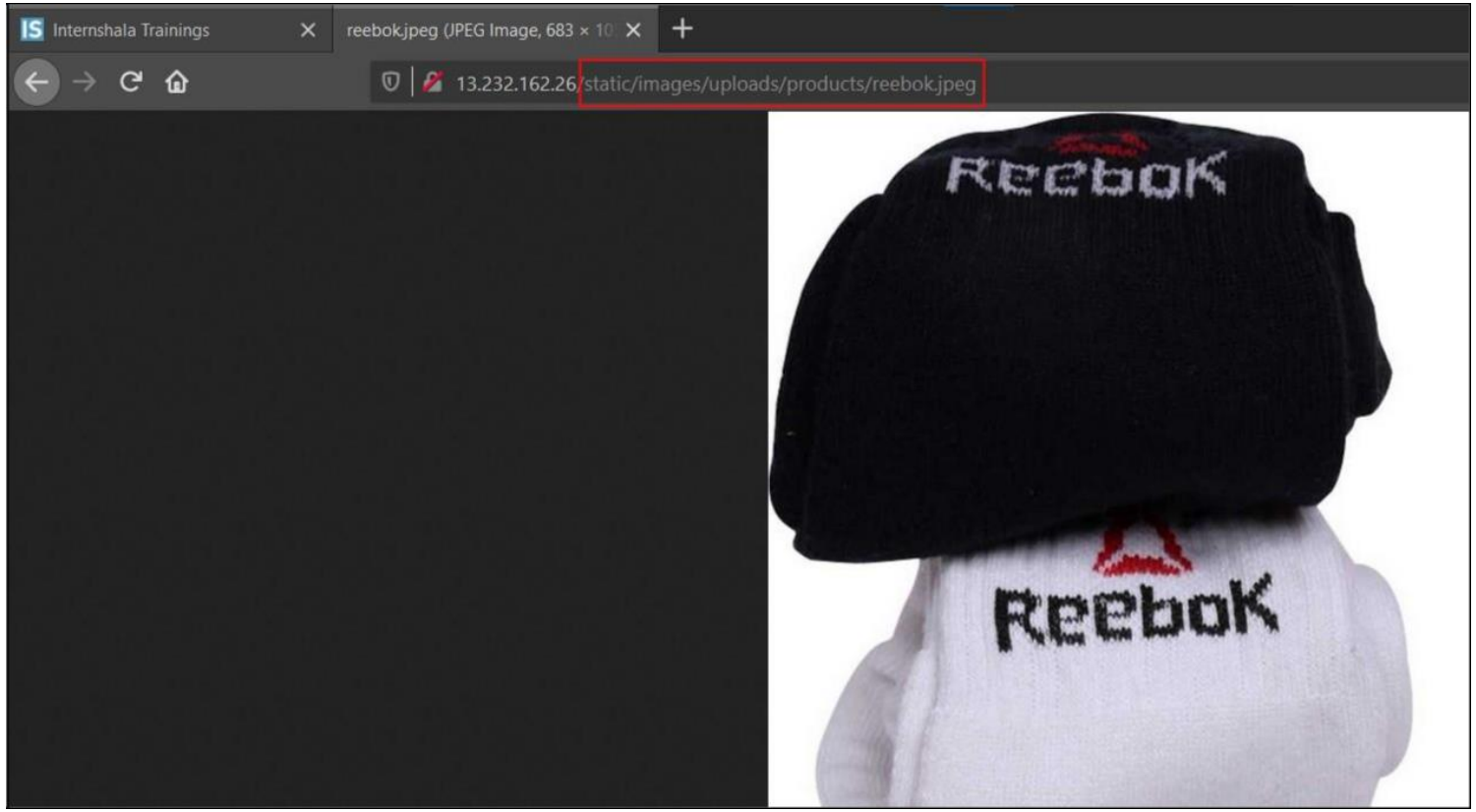
Other similar URLs that leaks critical information via directory listing vulnerability.

Affected URL:

- http://13.232.162.26/robots.txt

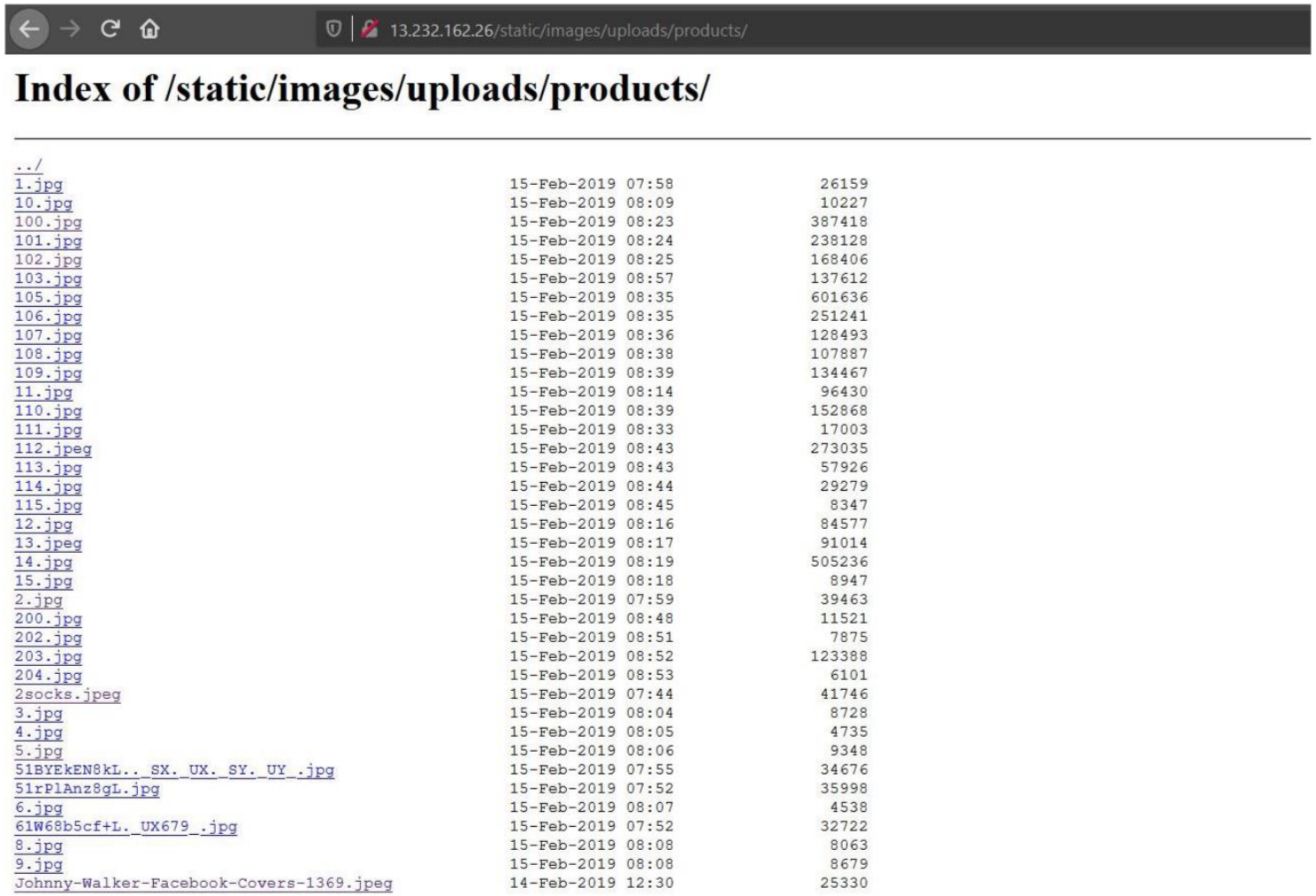
Observation

- Navigate to http://13.232.162.26/products.php
- Now, right click on the image of any product and then select View Image or you can even drag the image to a new tab.
- The page loads up as shown below, with the image of the selected product.\
- Notice the URL, it actually reveals the full path of the image.



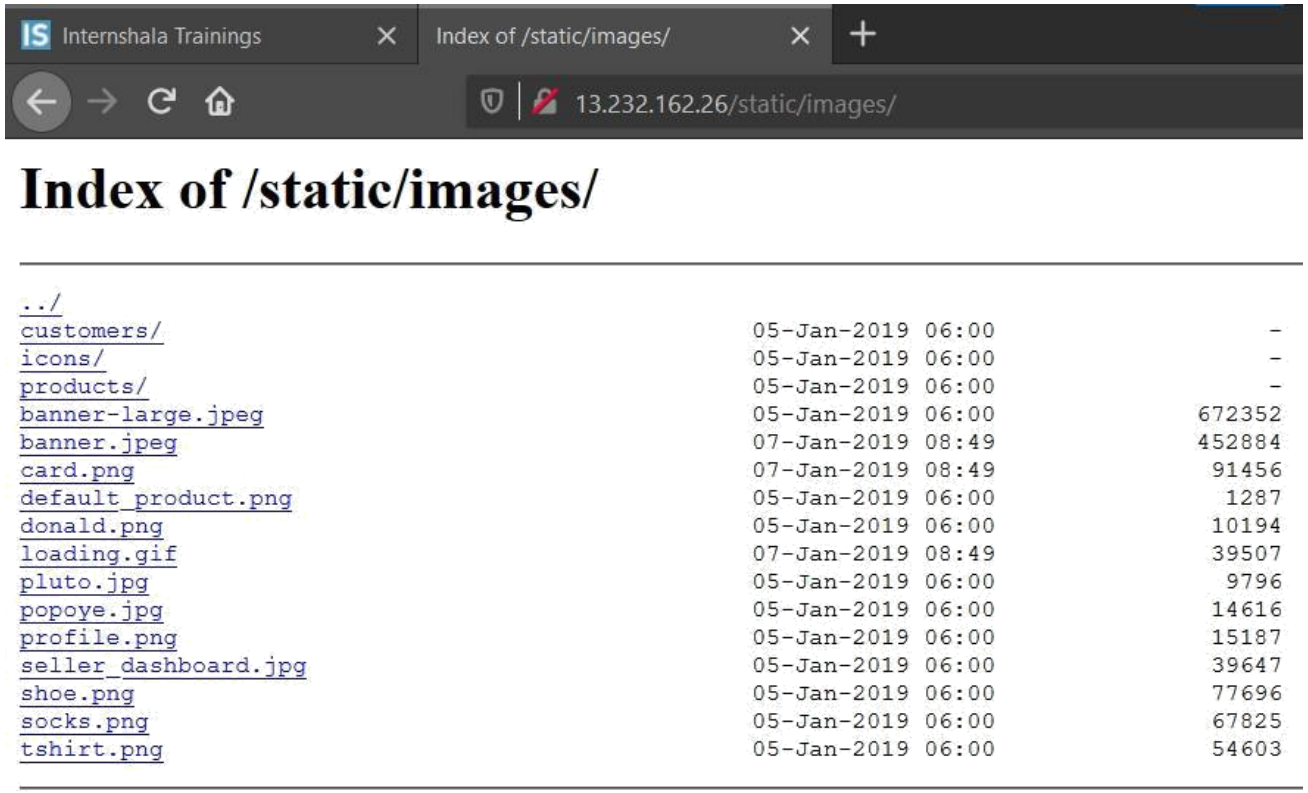
PoC – directory listings

- Now, if we remove the image name (here, reebok.jpeg) and hit enter.
- The following page with tons of information in it, will be displayed.



PoC – directory listings

- Navigate to http://13.232.162.26/static/images/
- Complete listing of directory is shown containing the images of all the customers along with the images of all the products in the website and also the administrator directory is also visible.



Recommendation

Take the following precautions:

- Two- Factor Authentication for sensitive data should be added with strong passwords.
- Find all PII stored and encrypt them with various techniques.
- Disable Directory Listing .
- Put an index.html in all folders with default message.

References

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

Below mentioned URL is vulnerable to personnel identifiable information leakage.

Affected URL :

- http://3.6.40.63/profile/16/edit/

13. PII Leakage

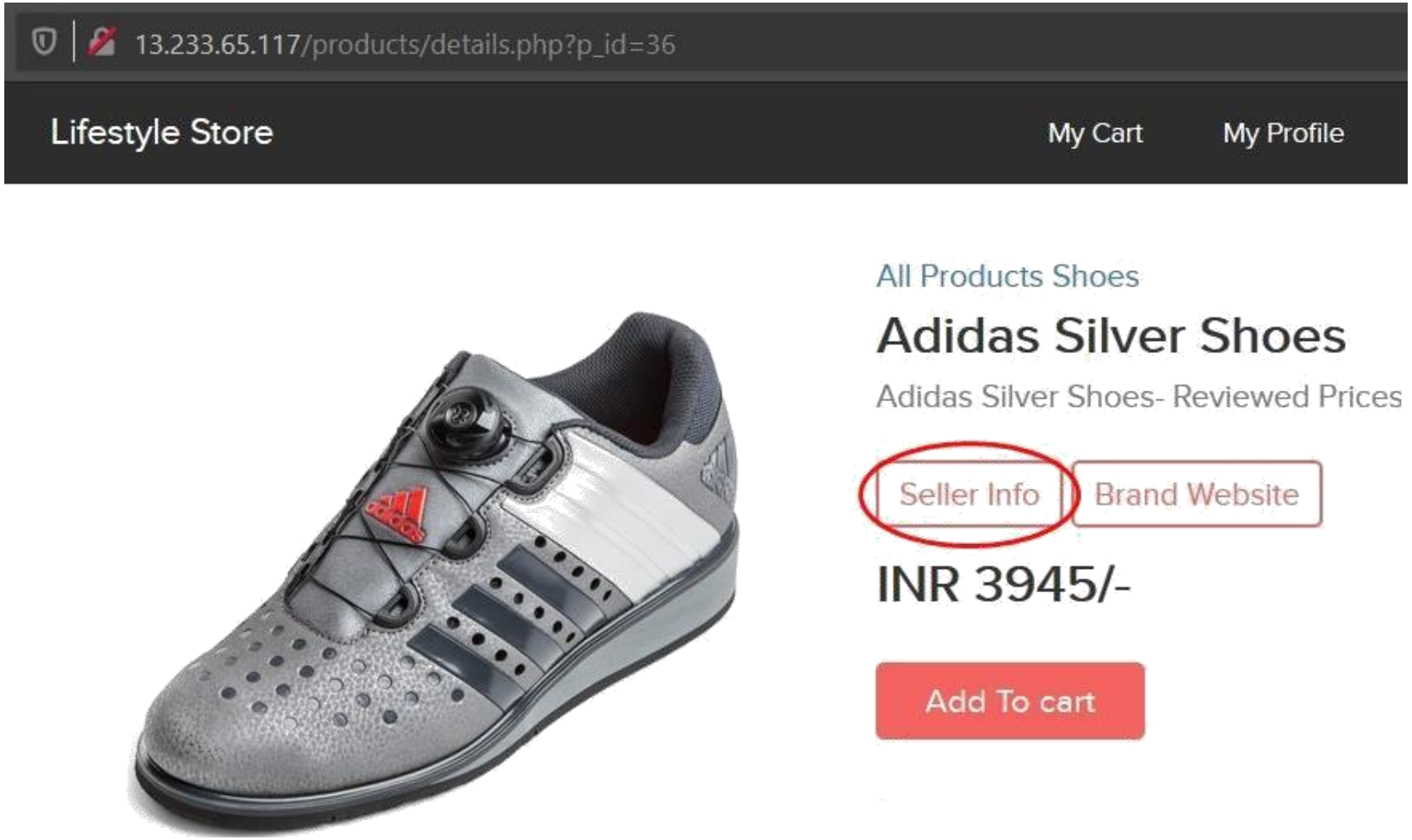
Below mentioned URL is vulnerable to personnel identifiable information leakage.

Affected URL :

http://3.6.40.63/profile/16/edit/

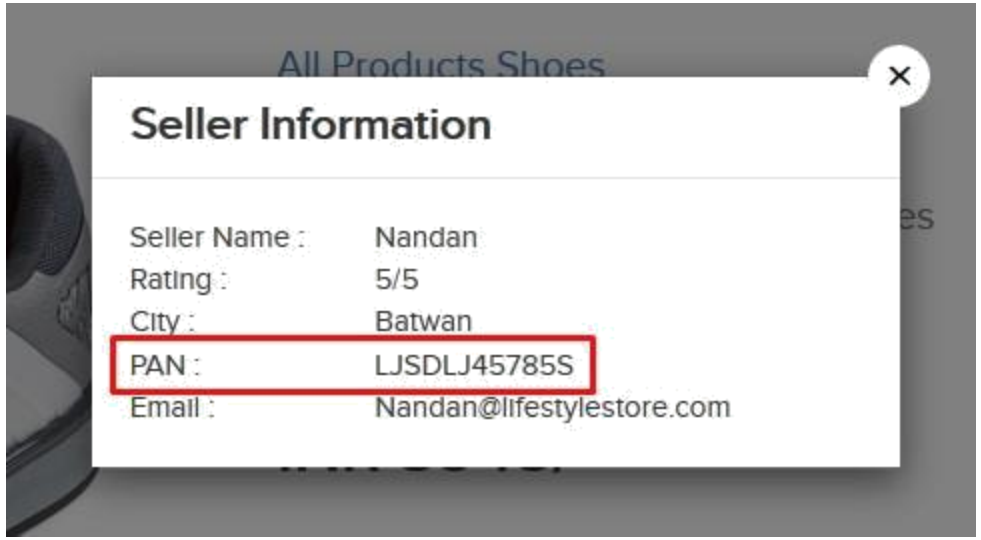
Observation

- Login to your account and go to Products page.
- In every product page the Seller Info is available, click on it.



PoC – pan card details are shown

- Upon clicking on Seller Info; Seller Name, Rating, City, Email along with PAN Card Details are shown.



Recommendation

- Hide critical information like the PAN Card details.
- Display only minimal required information about the sellers.

References

- https://www.imperva.com/learn/data-security/personally-identifiable-information-pii/
- https://hackerone.com/reports/374007

14. Open Redirection

Below mentioned URL is vulnerable to open redirection.

Affected URL :

- http://13.233.65.117/redirect.php?url=www.radhikafancystore.com

Affected Parameters :

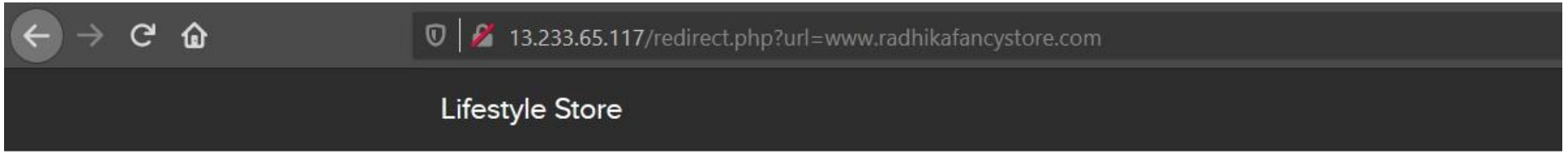
URL

Observation

- Login to your account and go to Products page.
- In every product page the Brand Website is available, click on it.

Observation

- Upon clicking on Brand Website, we are then being redirected to the brand’s website.

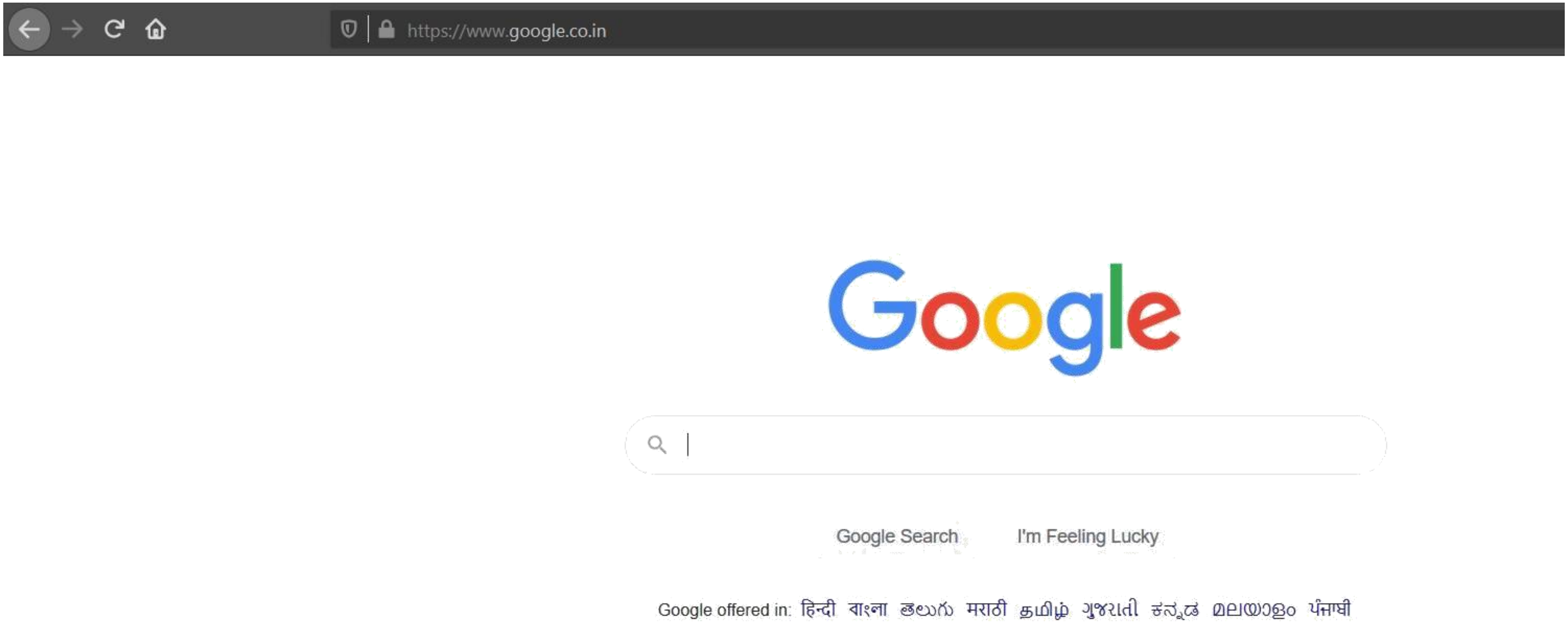


Observation

- Now, change the url from the brand website to some other website, here we use https://www.google.co.in/ and hit enter.

PoC – open redirection

- We have been redirected to the destination url.



Recommendation

- Check your Referrers.
- Design your app to avoid URL redirects or forwards as a best practice. If unavoidable, encrypt the target URL such that the URL:token mapping is validated on the server.
- Verify URL patterns using regular expressions to check if they belong to valid URLs. However, malicious URLs can pass that check.

References

- https://www.netsparker.com/blog/web-security/open-redirection-vulnerability-information-prevention/
- https://spanning.com/blog/open-redirection-vulnerability-web-based-application-security-part-1/
- https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/understanding-and-discovering-open-redirect-vulnerabilities/

Below mentioned URL is vulnerable to brute forcing and can be exploited for discounts.

Affected URL :

- http://15.207.106.113/cart/apply_coupon.php

15. Brute force Exploitation of Coupon Codes

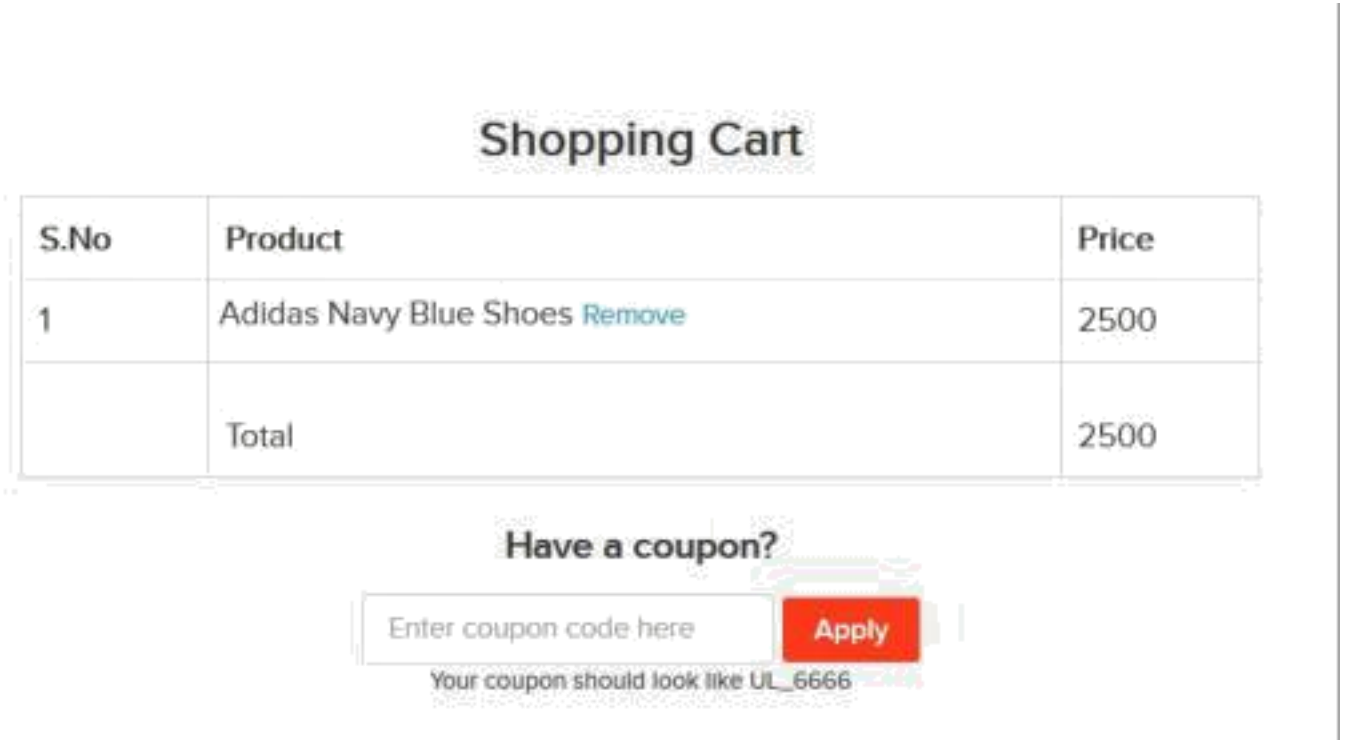
Below mentioned URL is vulnerable to brute forcing and can be exploited for discounts.

Affected URL :

- http://15.207.106.113/cart/apply_coupon.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

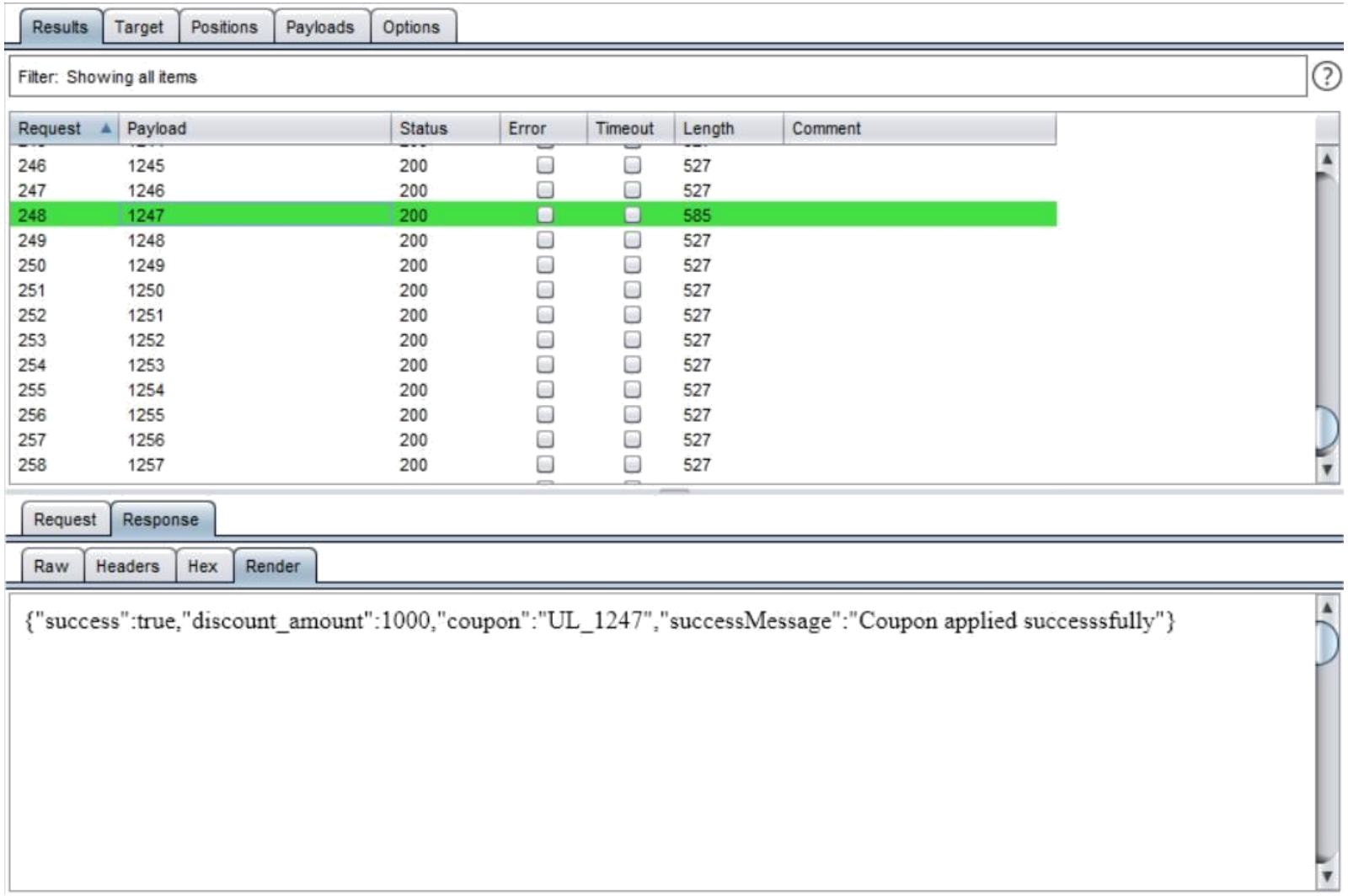
- Following request will be generated containing coupon code.

```
POST /cart/apply_coupon.php HTTP/1.1
Host: 15.207.106.113
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0
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: 92
Origin: http://15.207.106.113
DNT: 1
Connection: close
Referer: http://15.207.106.113/cart/cart.php
Cookie: key=552ABD04-CFD0-C7D1-748F-BC95609DB4BA; PHPSESSID=v7tsdb5m7nnm51co677neqmmr5; X-XSRF-TOKEN=593e631accdc7ea3fb8039bd89ede783314e5e73d762e1d0262886956070222c

coupon=UL_6666&X-XSRF-TOKEN=593e631accdc7ea3fb8039bd89ede783314e5e73d762e1d0262886956070222c
```

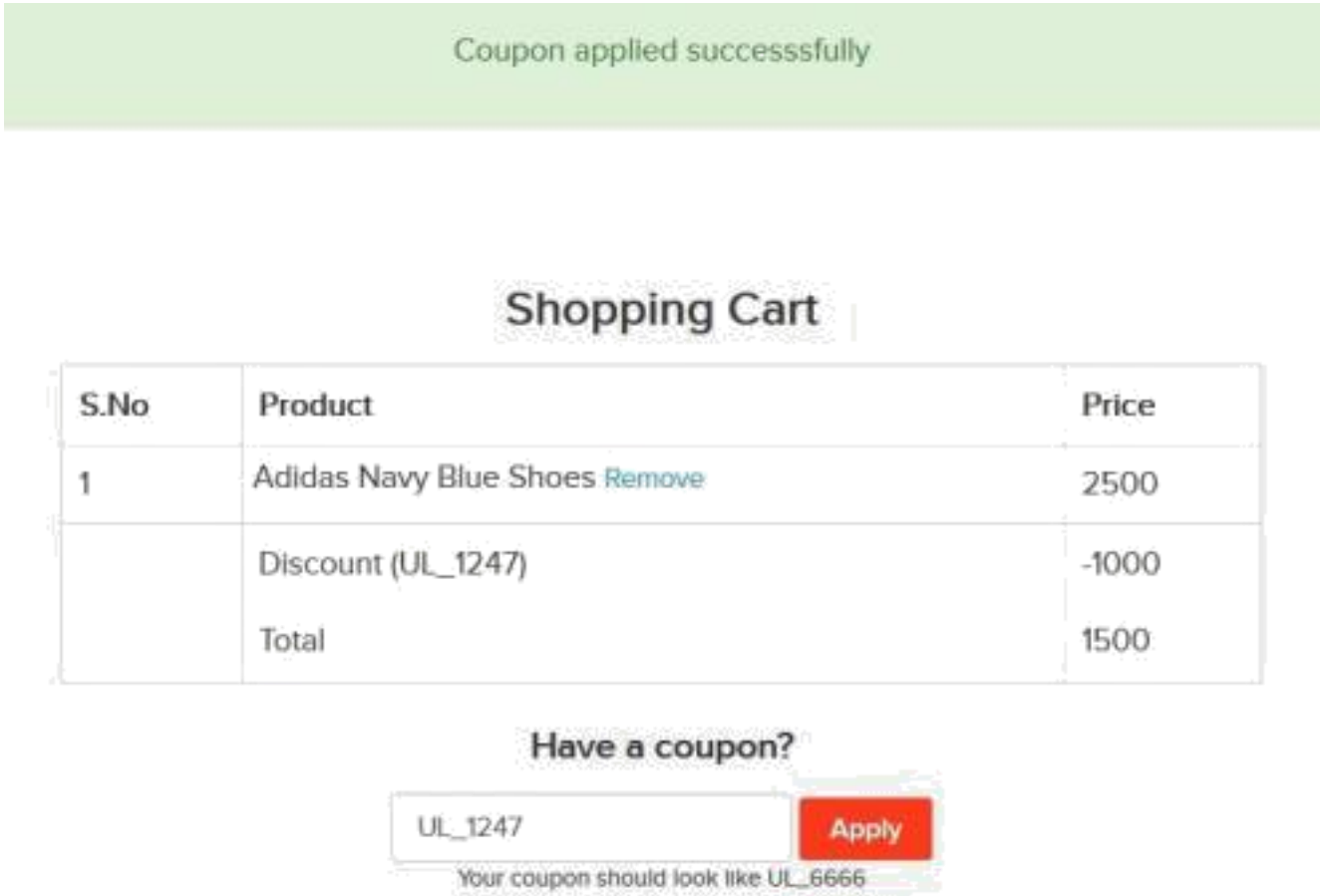
Observation

- 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.



PoC – coupon code applied successfully



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

Below mentioned URLs is vulnerable to command execution,

Affected URLs :

- http://13.233.65.117/wondercms/files/b374kmini.php
- http://13.127.150.195/admin31/console.php

16. Command Execution Vulnerability

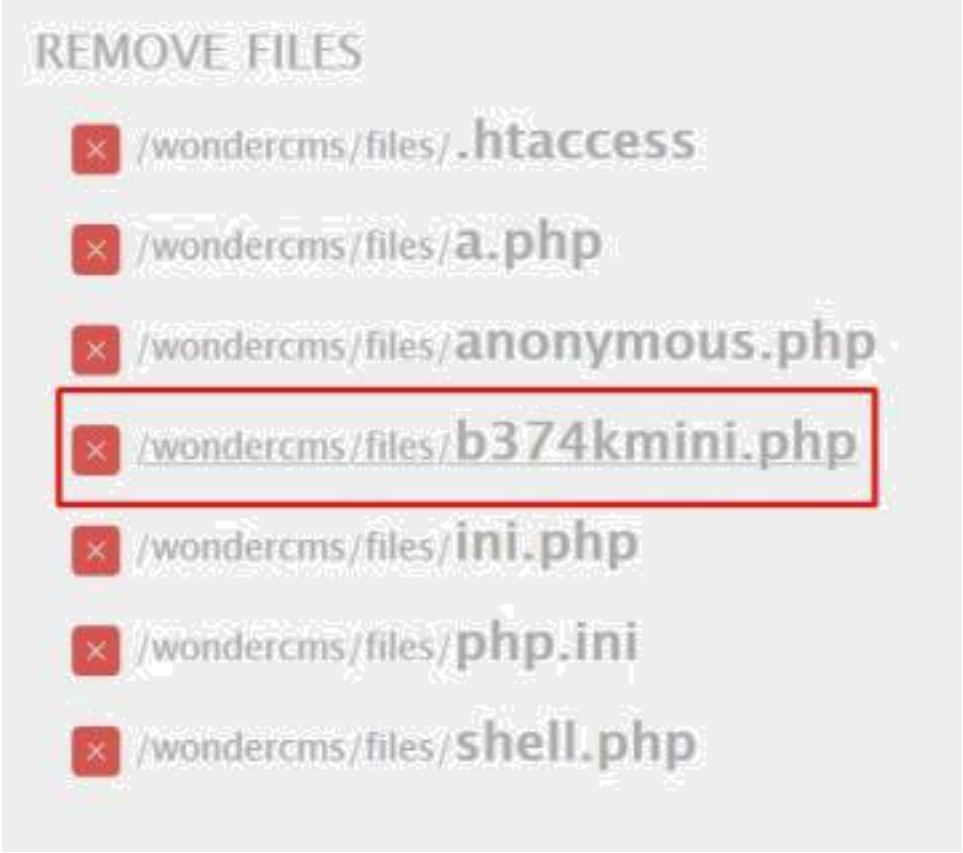
Below mentioned URLs is vulnerable to command execution,

Affected URLs :

- http://13.233.65.117/wondercms/files/b374kmini.php
  http://13.127.150.195/admin31/console.php

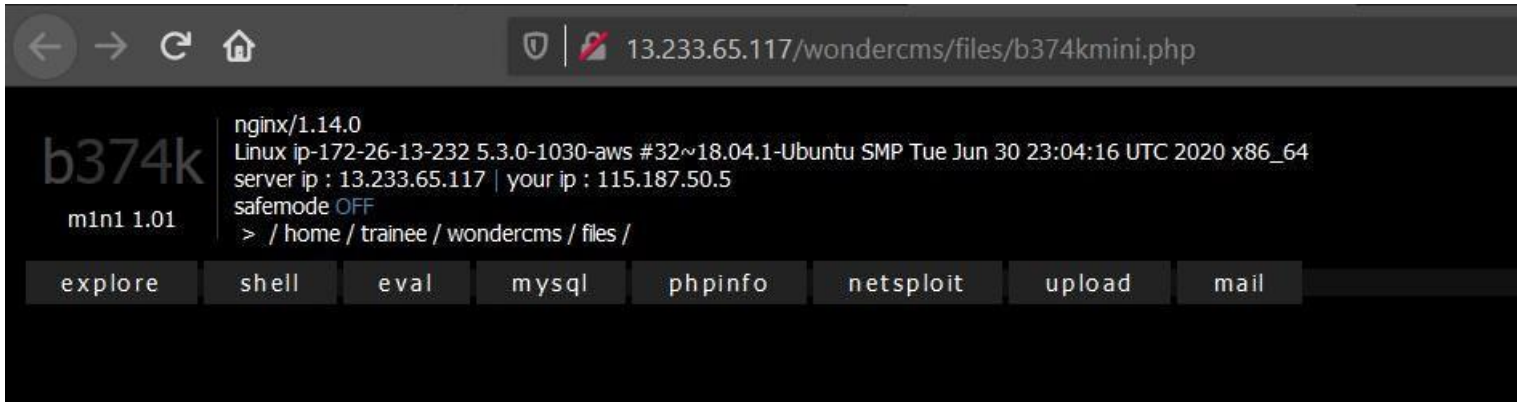
Observation

- 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

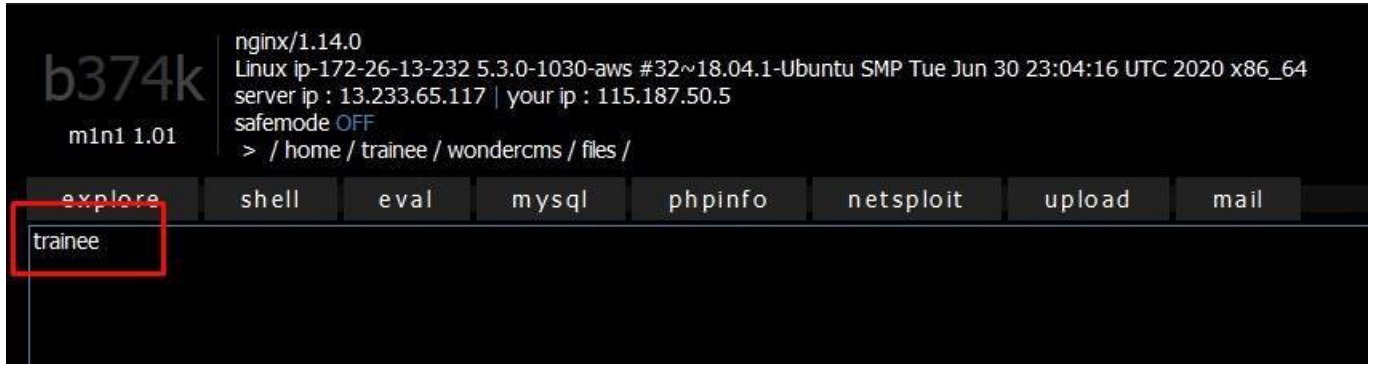
- It looks like, this is a small and simple PHP-shell that has an explorer, allows shell command execution, mysql queries, and more.



PoC – command execution

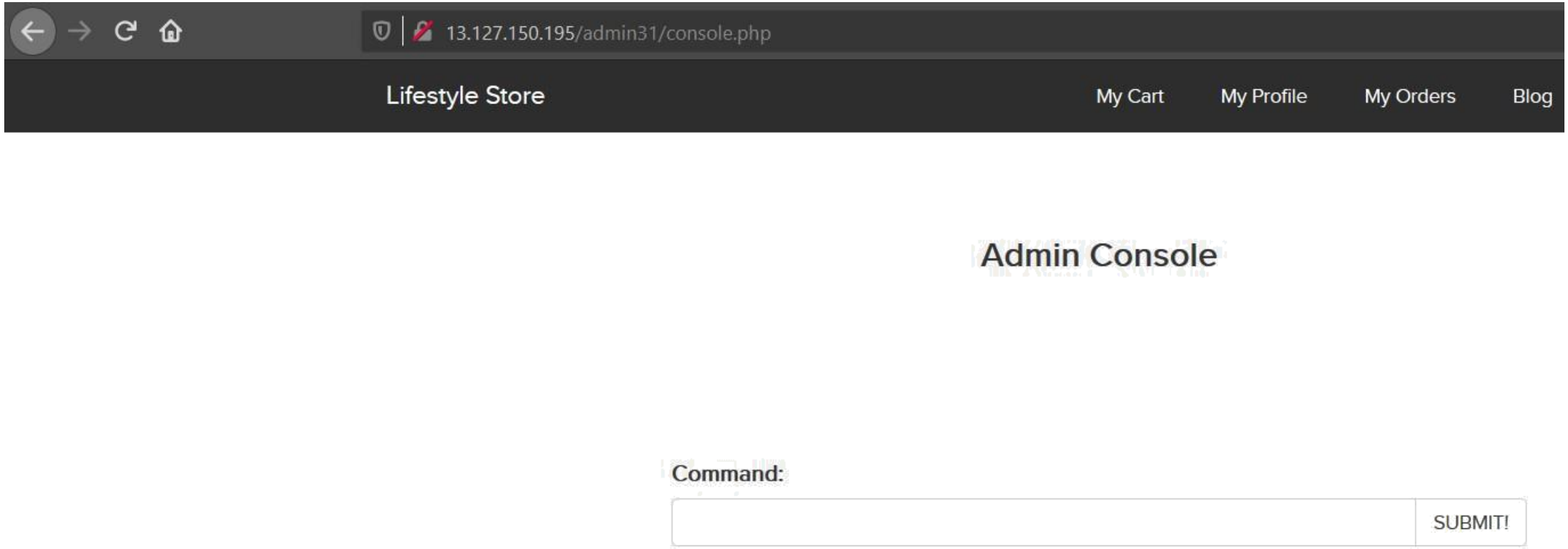
- Type in the Command: whoami and press Go!

- The command was executed successfully.



Observation

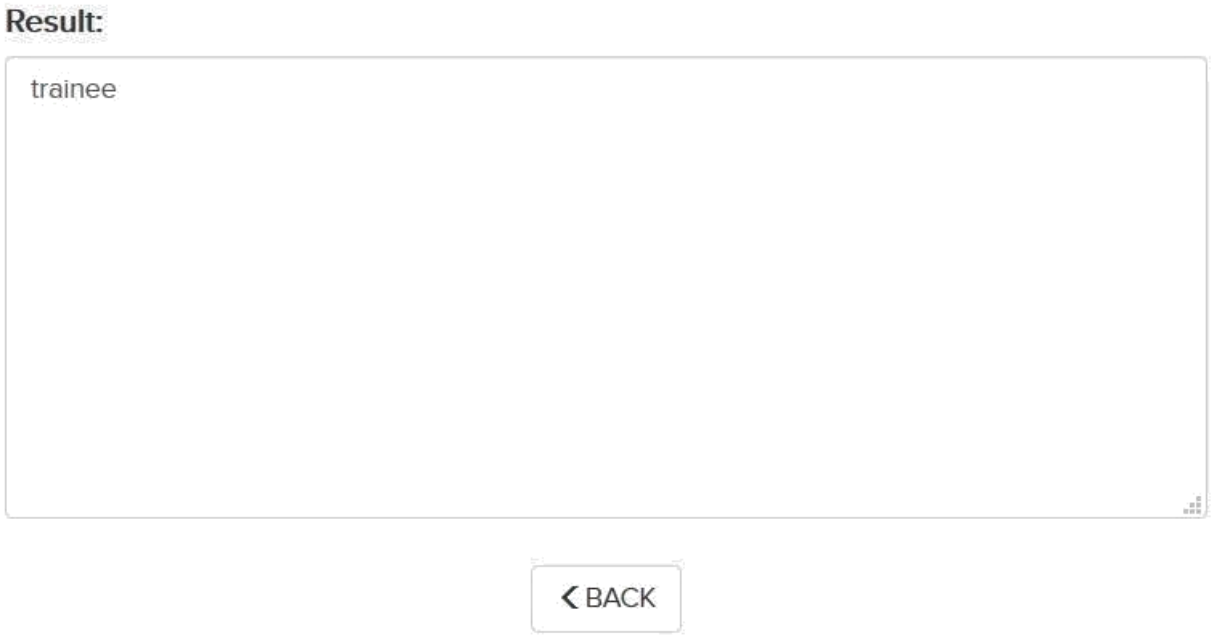
- As a customer, Login to your account.
- Now, forcefully type in the url for going to the admin console http://13.127.150.195/admin31/console.php (you came to know about this url while testing vulnerabilities for Vulnerability Report No. 4, Rate Limiting Flaws), and press enter.



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.



Recommendation

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

References

17. Forced Browsing

Below mentioned URLs is vulnerable to forced browsing.

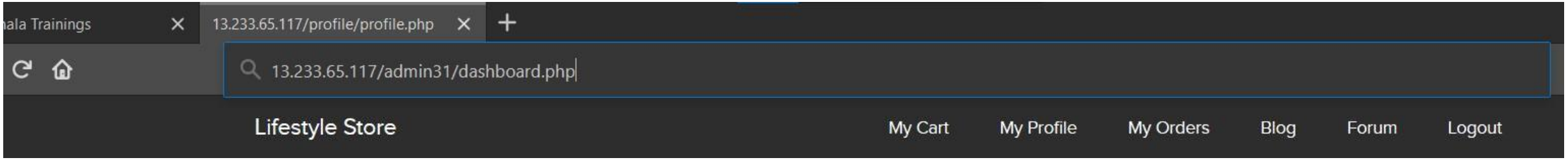
Affected URL :
http://13.233.24.9/

Forced URLs :

- http://13.233.65.117/admin31/dashboard.php
- http://13.127.150.195/admin31/console.php

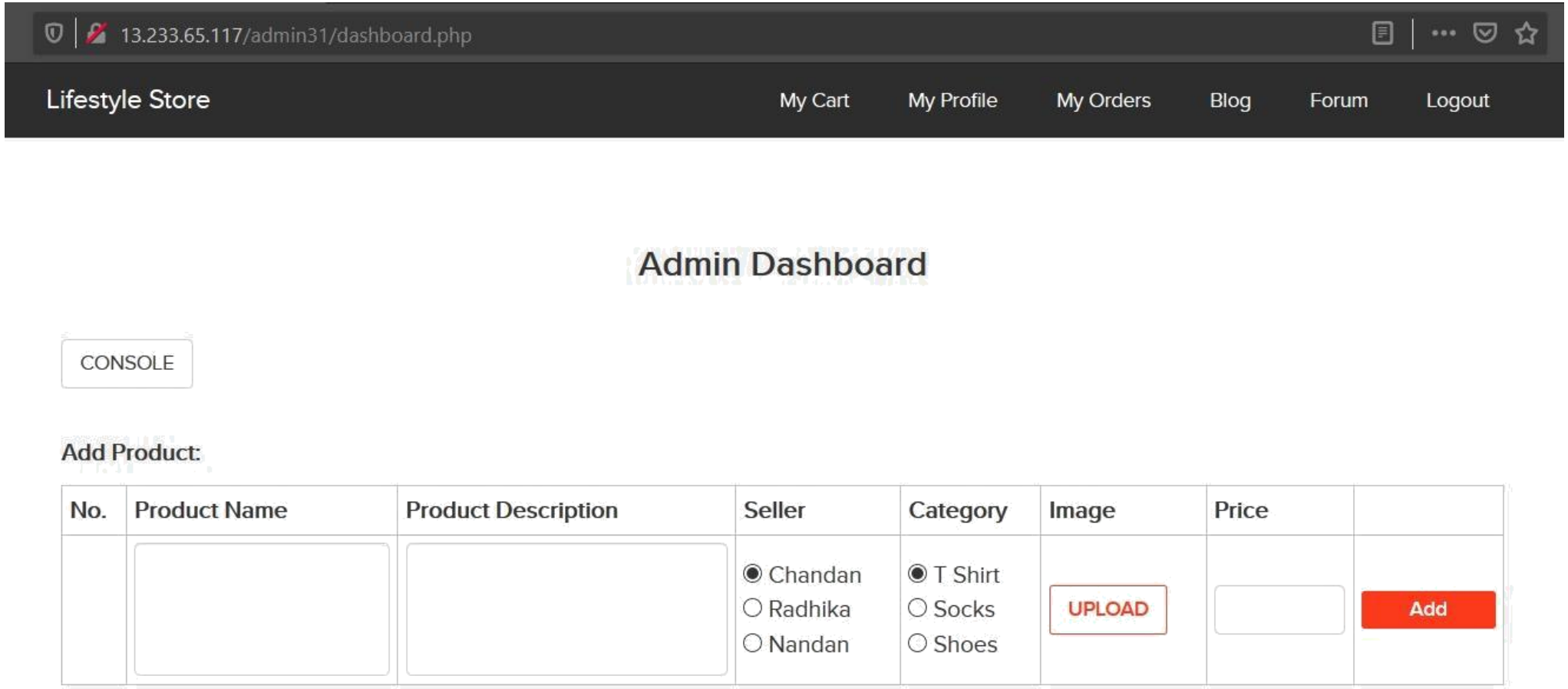
Observation

- As a customer, Login to your account.
- Now, forcefully type in the url for going to the admin dashboard http://13.233.65.117/admin31/dashboard.php (you came to know about this url while testing vulnerabilities for Vulnerability Report No. 4, Rate Limiting Flaws).



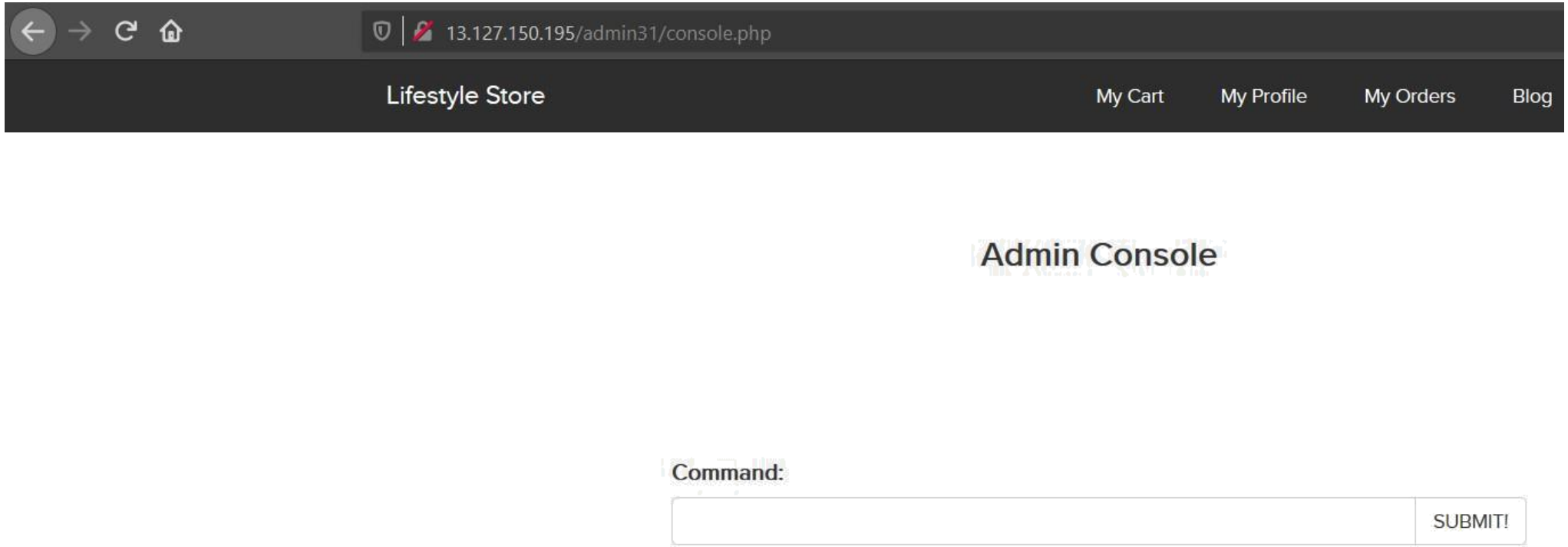
PoC – admin dashboard access

- 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.



ecommendation

- 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/

Below mentioned URLs are vulnerable to cross-site request forgery.

Affected URLs :

- http://13.233.24.9/profile/change_password.php
- http://13.233.24.9/cart/cart.php

18. Cross-Site Request Forgery

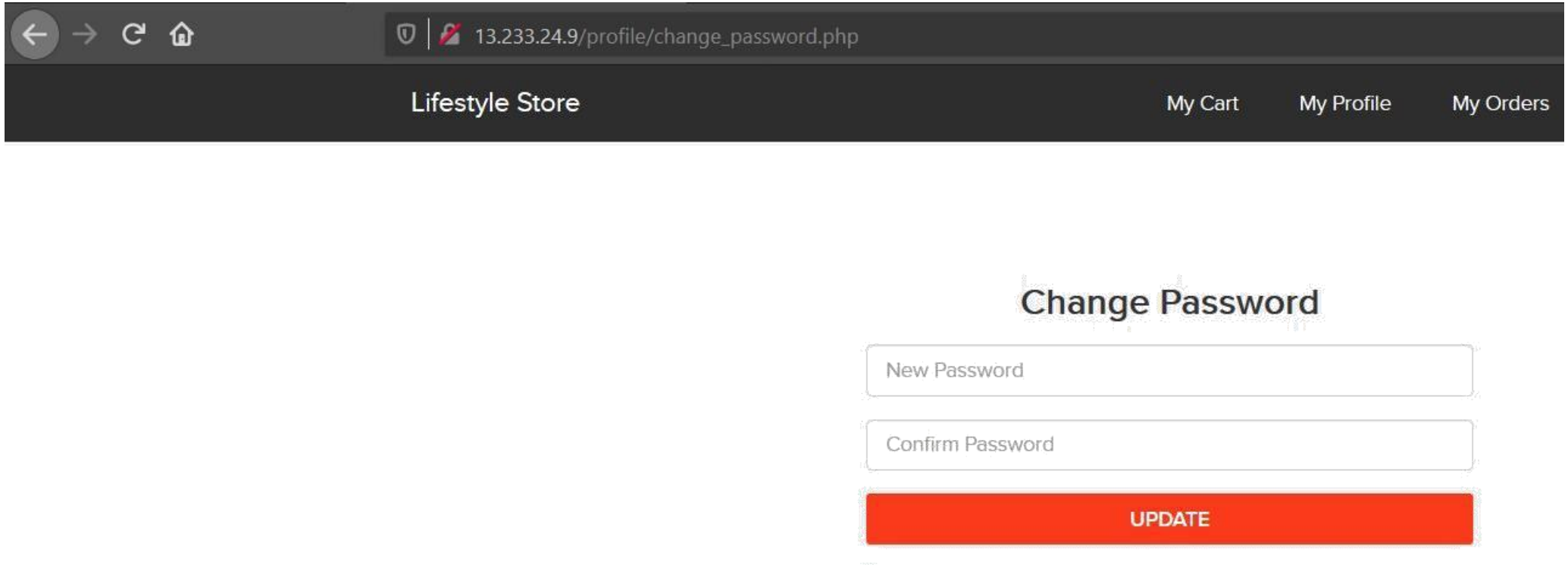
Below mentioned URLs are vulnerable to cross-site request forgery.

Affected URLs :

- http://13.233.24.9/profile/change_password.php
- http://13.233.24.9/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>
```

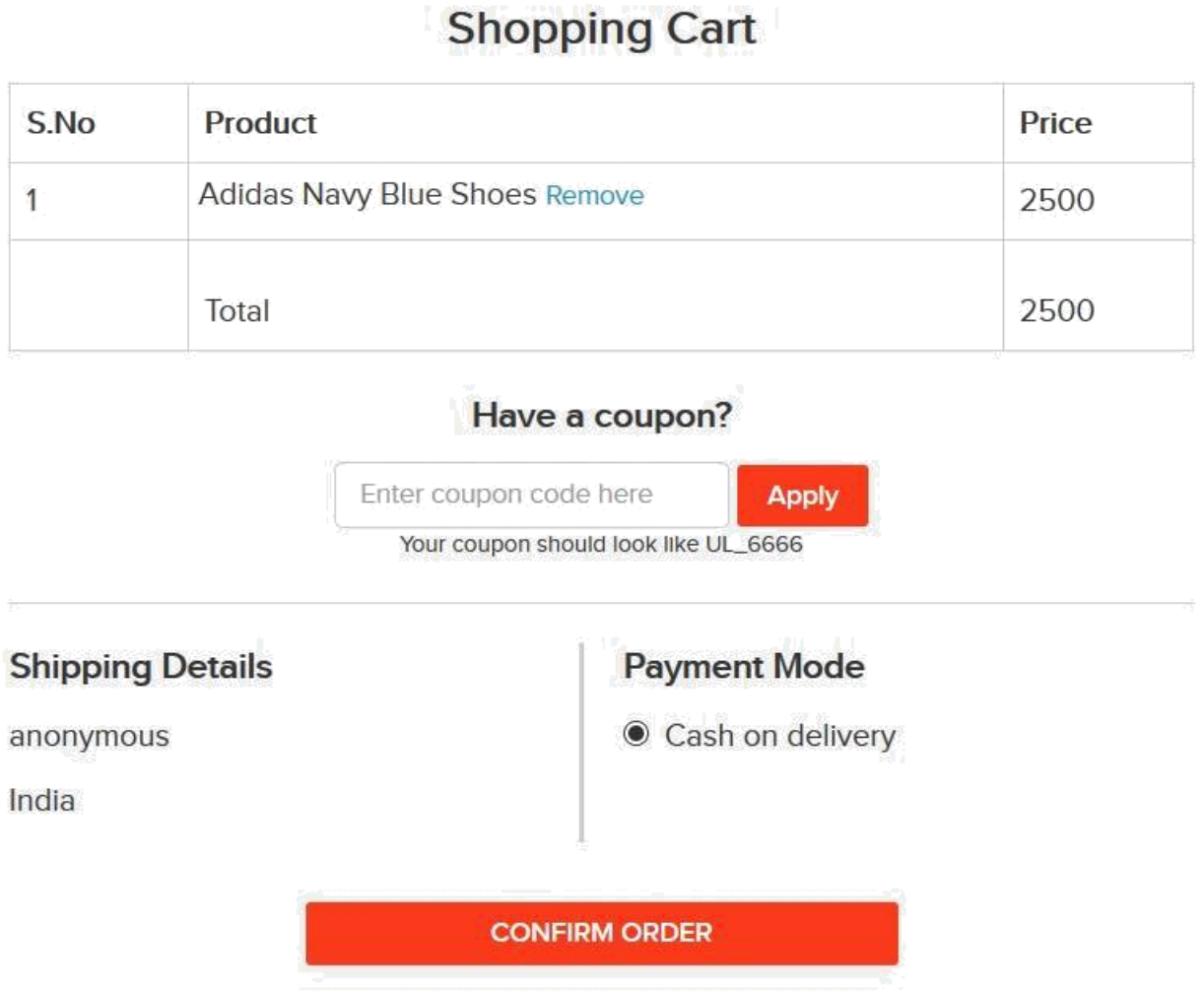
- Type in a new set of password and click on Update button, upon clicking on it, we get a Success Message.

{"success":true,"successMessage":"Password updated succesfully."}

- 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.



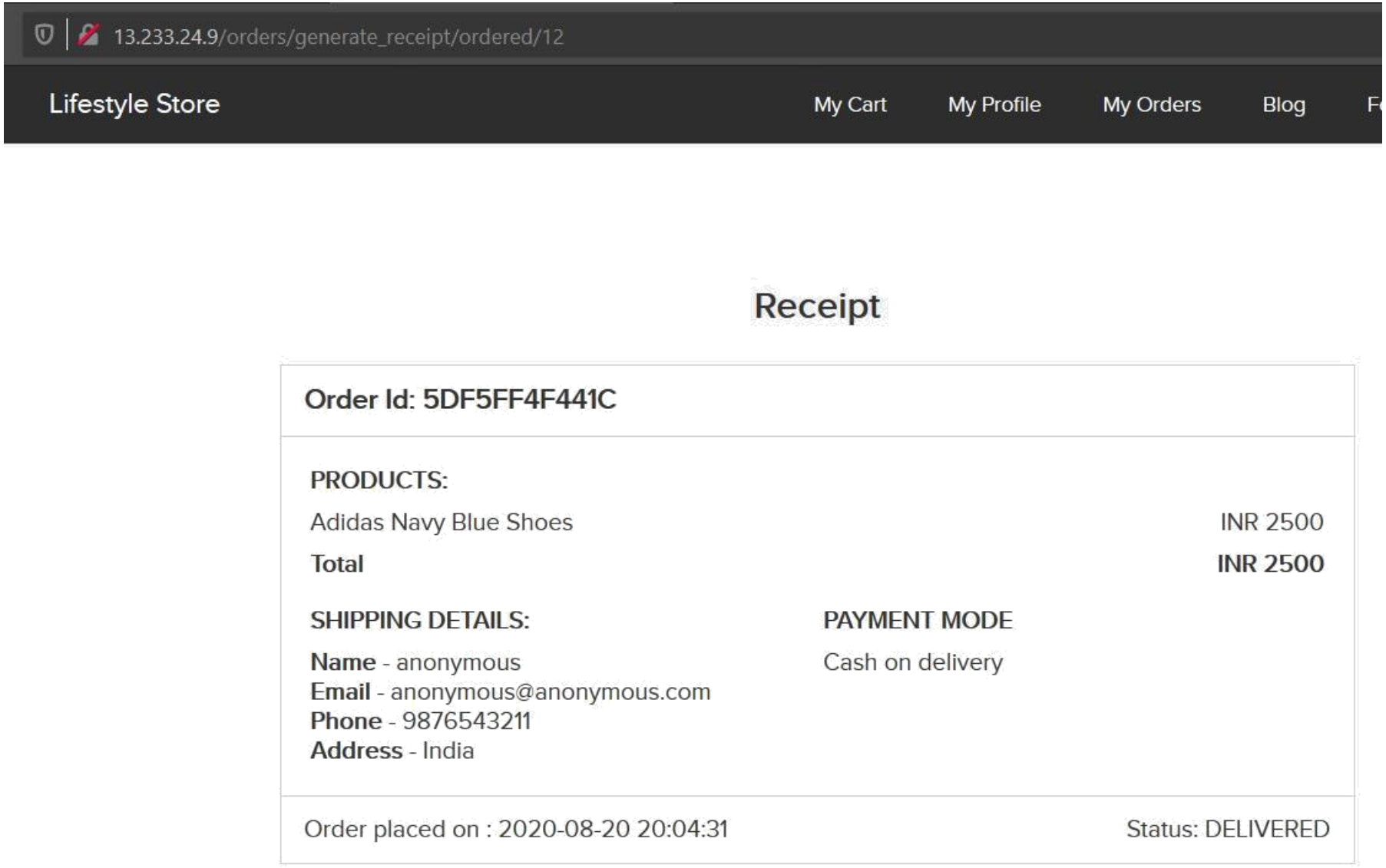
PoC – order confirmed successfully

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

```
<html>

<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.



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

Below mentioned URL shows the seller accounts and passwords.

Affected URL :

- http://15.206.159.87/userlist.txt

19. Seller Account Access

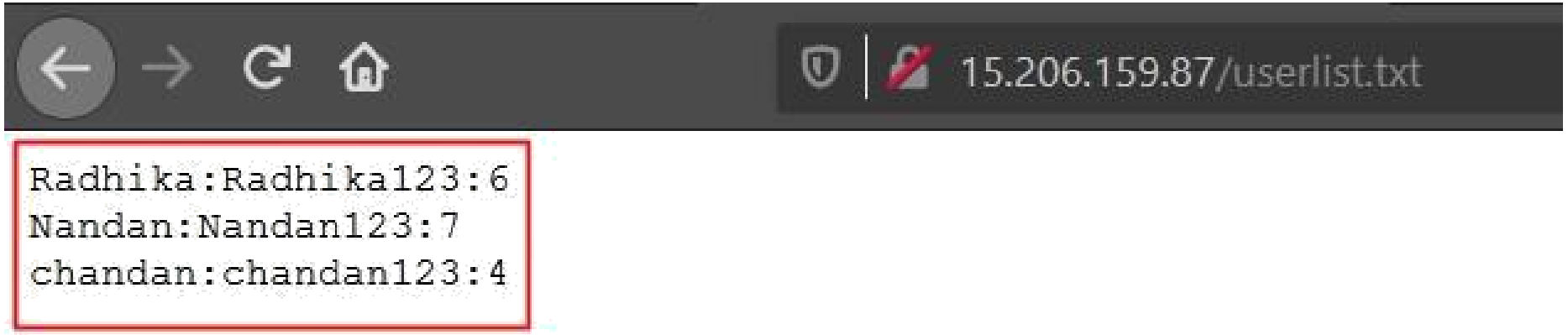
Below mentioned URL shows the seller accounts and passwords.

Affected URL :

- http://15.206.159.87/userlist.txt

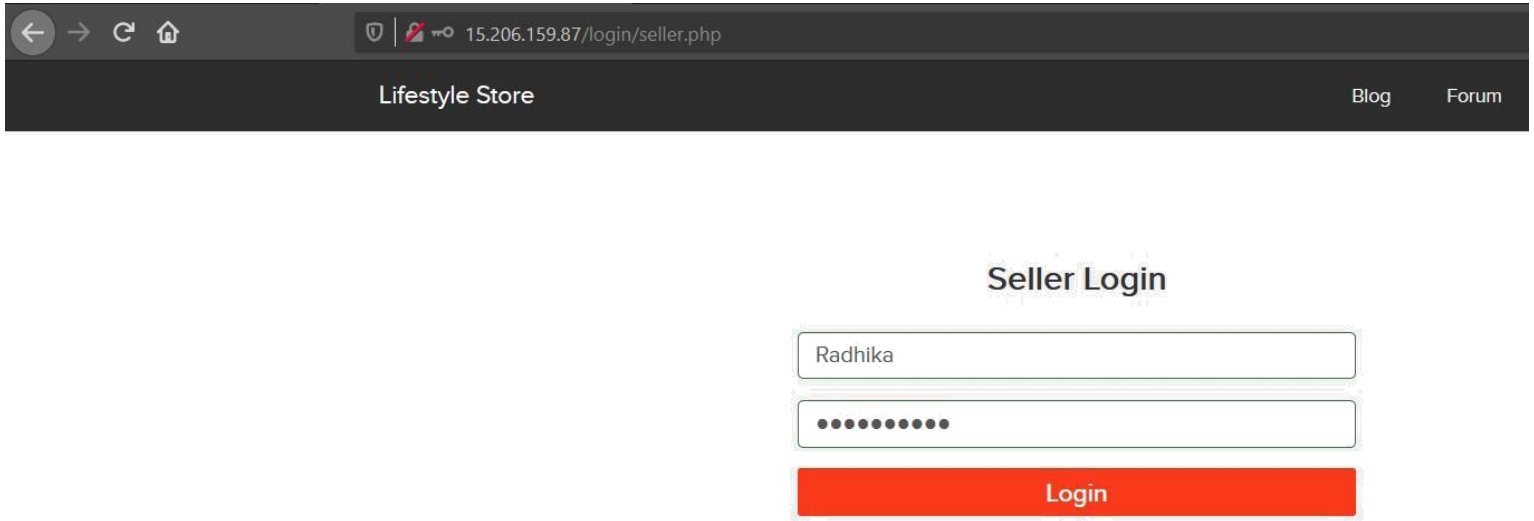
Observation

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

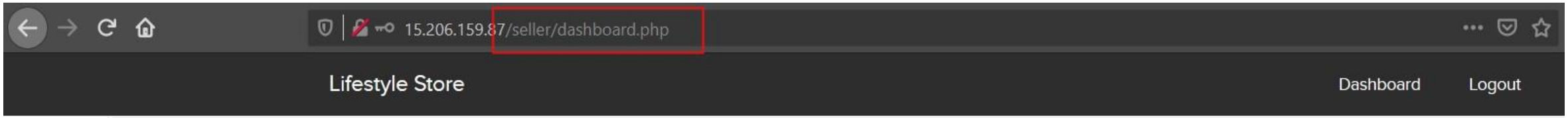


PoC - attacker has the seller dashboard access

- 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


15.206.159.87/seller/dashboard.php
Lifestyle Store
Dashboard
Logout

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