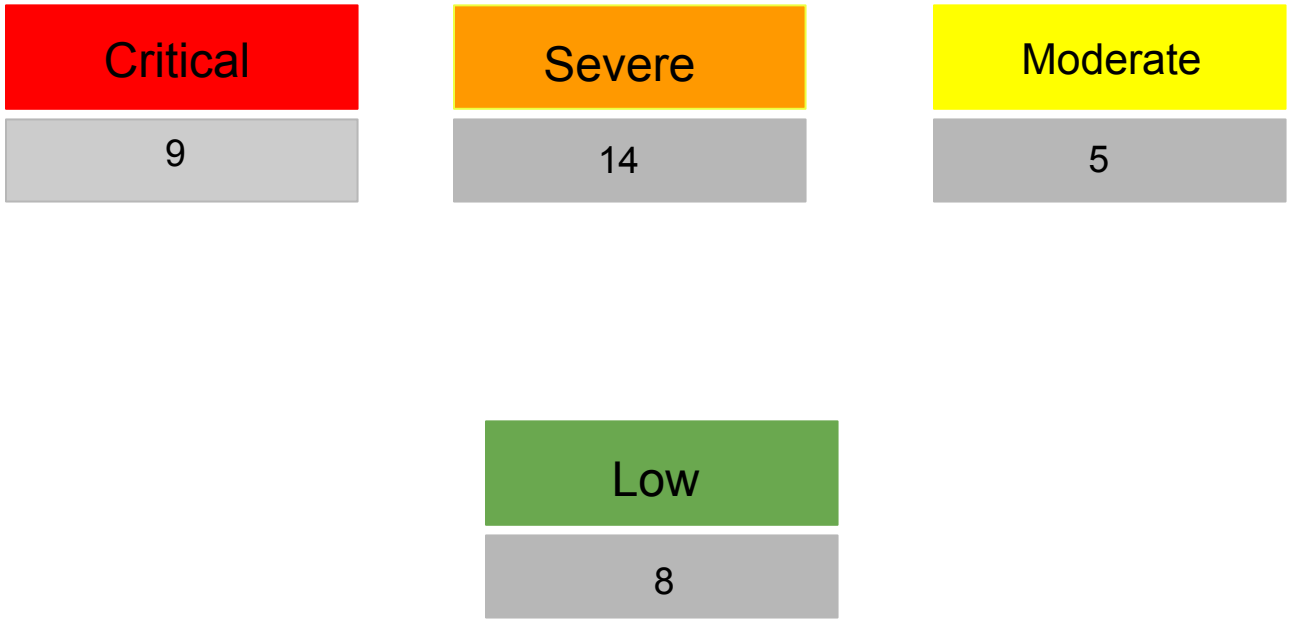
Lifestyle Store
Detailed Developer Report

Security Status – Extremely Vulnerable

1. The hacker can steal all the data from database of lifestyle store (SQLi).
2. Hacker can control the server and steal the critical information in the database by shell upload.
3. Hacker can access entire content of website(Gaining admin access).
4. Hacker can extract critical information of all customers by user_id (IDOR).
5. Hacker can change the password and gain the access of admin account by OTP Bypass.
6. Hacker can gain access of any account like admin account, seller account by forced browsing.
7. Hacker run the command in admin account and get the information about the server and system.

Vulnerability Statistics



Vulnerabilities

No.	Severity	Vulnerability	Count
1	Critical	SQL Injection	1
2	Critical	Arbitary File Upload	1
3	Critical	Access to admin panel	1
4	Critical	Unauthorised access to customer details(IDOR)	3
5	Critical	Reset password of admin by OTP Bypass	1
6	Critical	Forced Browsing	1
7	Critical	Run Command in admin panel	1
8	Severe	Cross Site Request Forgery	3
9	Severe	Default/Weak Password	2
10	Severe	Cross Site Scripting	3
11	Severe	Rate Limiting Flaw	4
12	Severe	Open Redirection	1
13	Severe	Crypto Configuration Flaw	1

No.	Severity	Vulnerability	Count
14	Moderate	Directory Listing	1
15	Moderate	Personaly Identifiable Information(PII) leakage	1
16	Moderate	Outdated version of using components	2
17	Moderate	Unrequired information of seller	1
18	Low	Server side misconfiguration flaw	2
19	Low	Descriptive error messages	2
20	Low	Default files/pages	4

1) SQL Injection

1) SQL Injection (Critical)

Below mention URL is vulnerable to SQL injection.

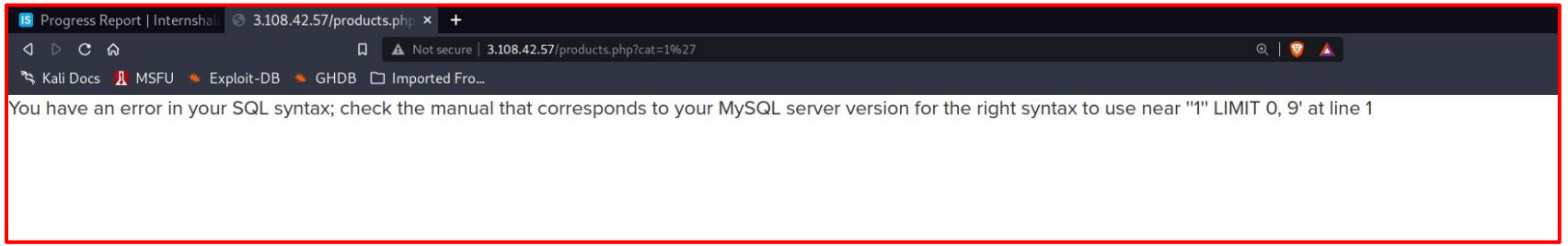
Affected URL :
http://url.com/products.php?cat=1

Affected Parameters :
cat (GET Parameter)

Payload :
cat=1’

Observation

Below you see the category 1 for T-shirts. Click on the socks the category will be change into a cat=2 that is called the parameter of this URL is GET based parameter. Then we apply the single quote in the parameter ‘cat’ . We get the complete SQL syntax error from the server.



Observation

We then put --+ :products.php?cat1'-- + and the SQL syntax error gets removed which shows that it is vulnerable to SQL Injection.
Using automated tool **sqlmap**, we find the SQL injection vulnerabilities in this application.

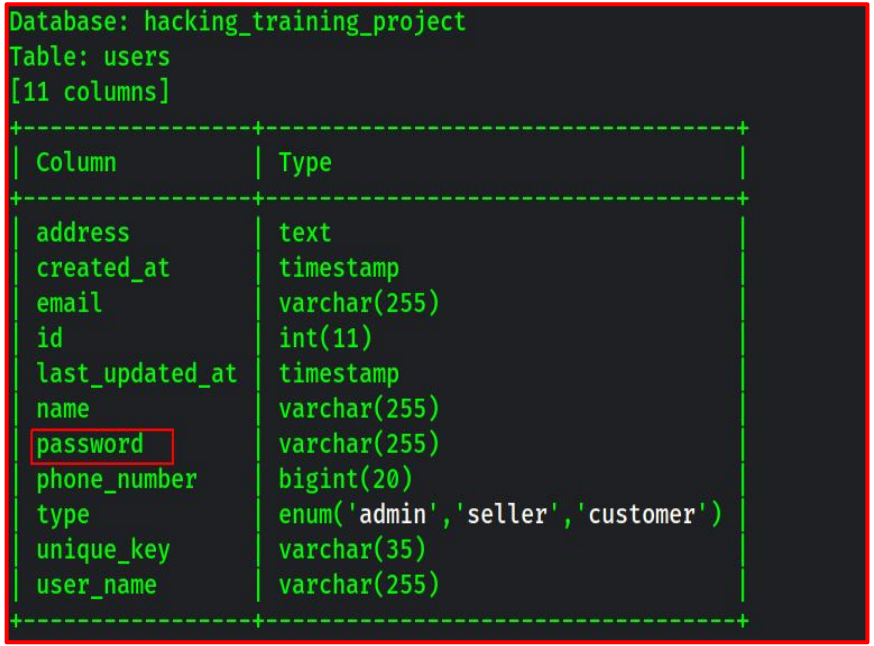
Sqlmap command :-
Sqlmap -u http://url.com/products.php?cat=1 --cookie "key=XXXXXXXXXXXXXXX" --dbs -T "users" -C name,password --dump

Proof of Concept (PoC) : Attacker can dump arbitrary data like No of databases ,username, **passwords**

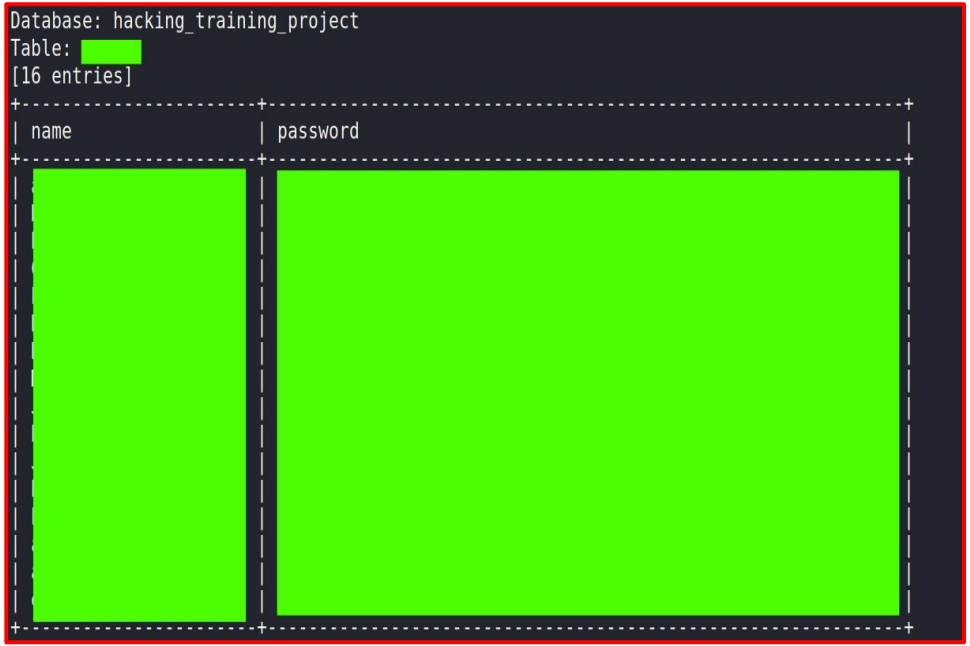
```
18:31:32] [INFO] fetching database names
available databases [2]:
*] hacking_training_project
*] information_schema
```

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

Proof of Concept (PoC) :



```
Database: hacking_training_project
Table: users
[11 columns]
+-----------------+-----------------------------------+
| Column          | Type                              |
+-----------------+-----------------------------------+
| address         | text                              |
| created_at      | timestamp                         |
| email           | varchar(255)                      |
| id              | int(11)                           |
| last_updated_at | timestamp                         |
| name            | varchar(255)                      |
| password        | varchar(255)                      |
| phone_number    | bigint(20)                        |
| type            | enum('admin','seller','customer') |
| unique_key      | varchar(35)                       |
| user_name       | varchar(255)                      |
+-----------------+-----------------------------------+
```



```
Database: hacking_training_project
Table: [redacted]
[16 entries]
+----------------------+-----------------------------------------------------------+
| name                 | password                                                  |
+----------------------+-----------------------------------------------------------+
| [redacted]           | [redacted]                                                |
+----------------------+-----------------------------------------------------------+
```

Business Impact – Extremely High

1. 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.
2. Previous slide has the screenshot of users table which shows user credentials being leaked that too in very weak encryption technique.
3. 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.

RECOMMENDATIONS

1. Whitelist User Input: Whitelist all user input for expected data only. For example if you are expecting a flower name, limit it to alphabets only upto 20 characters in length. If you are expecting some ID, restrict it to numbers only.
2. Prepared Statements: Use SQL prepared statements available in all web development languages and frameworks to avoid attacker being able to modify SQL query
3. 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.
4. Do not run Database Service as admin/root user
5. Disable/remove default accounts, passwords and databases
6. 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) Arbitrary File Upload

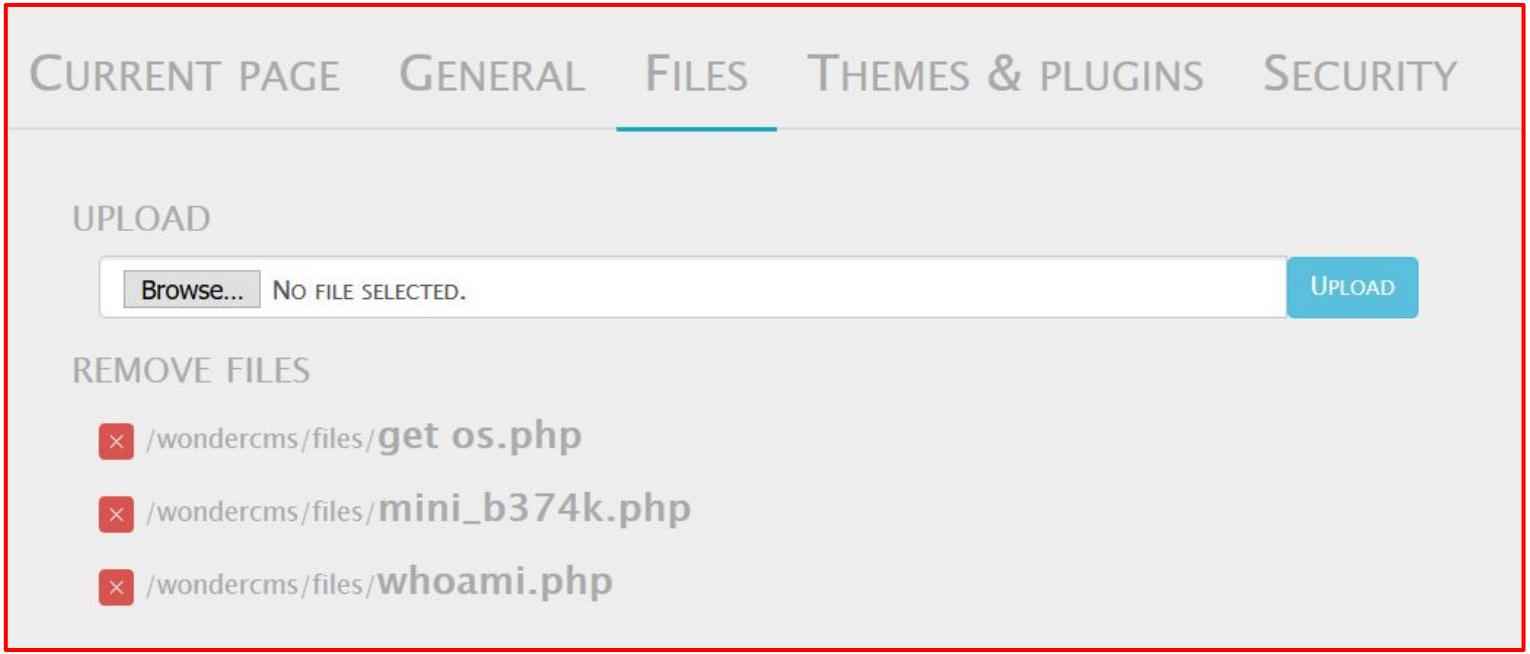
2) Arbitrary File Upload (Critical)

Below mentioned URL is vulnerable to Arbitrary File Upload and making other admin level changes

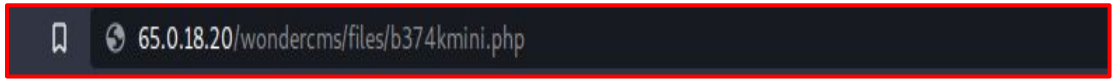
Affected URL :
•http://65.0.18.20/wondercms

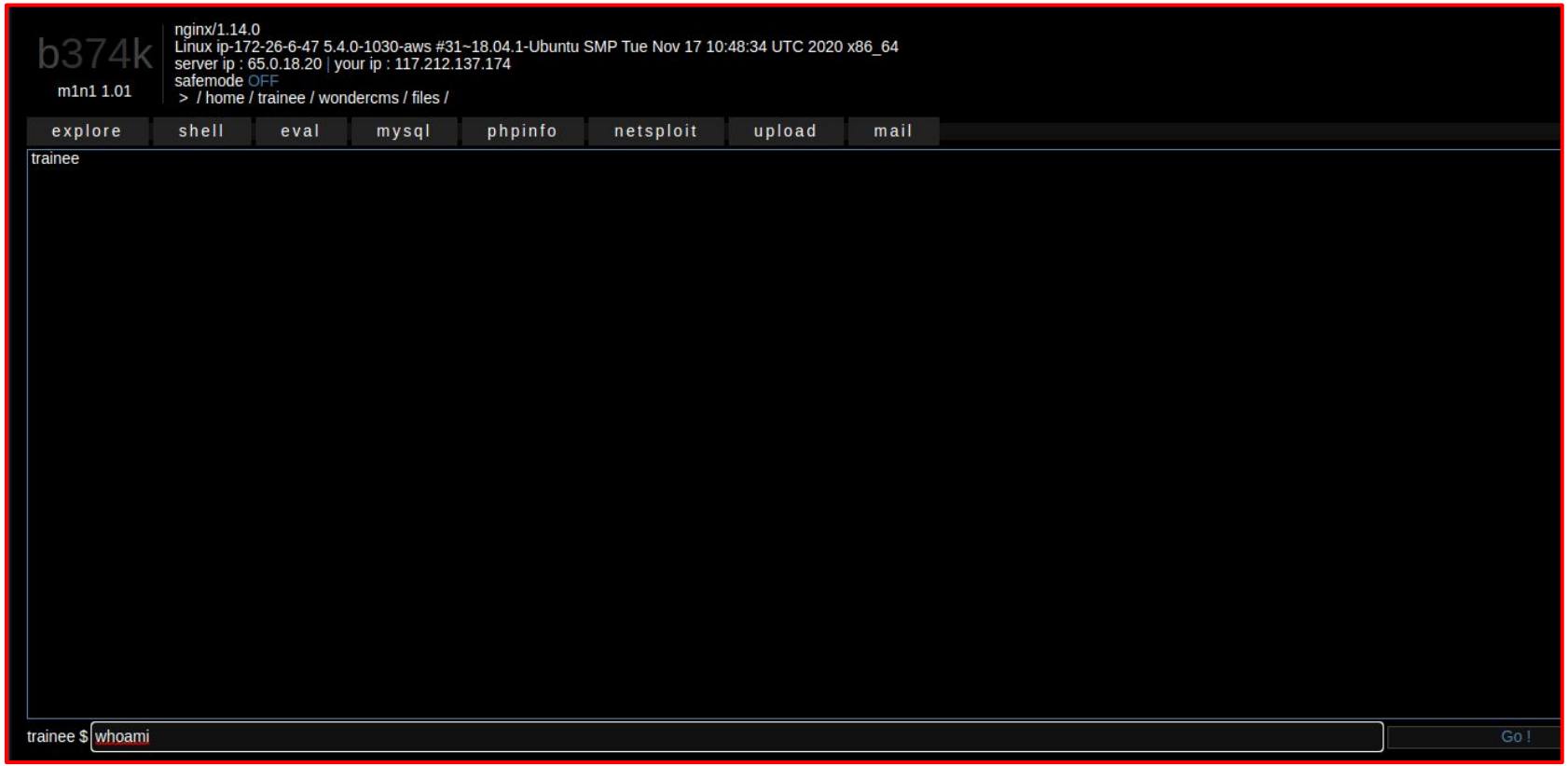
Observation :-

When we login to http://65.0.18.20/wondercms/ then in settings file uploads we can upload any arbitrary php file like reverse shell. Server totally fails to check whether the php file is malicious or not.



Proof of Concept (POC)





Business Impact – Extremely High

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

1. Important files
2. Password
3. And much more...
4. 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

1. Change the Admin password to something strong and not guessable.
2. 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.opswat.com/blog/file-upload-protection-best-practices

3) Access to Admin Panel

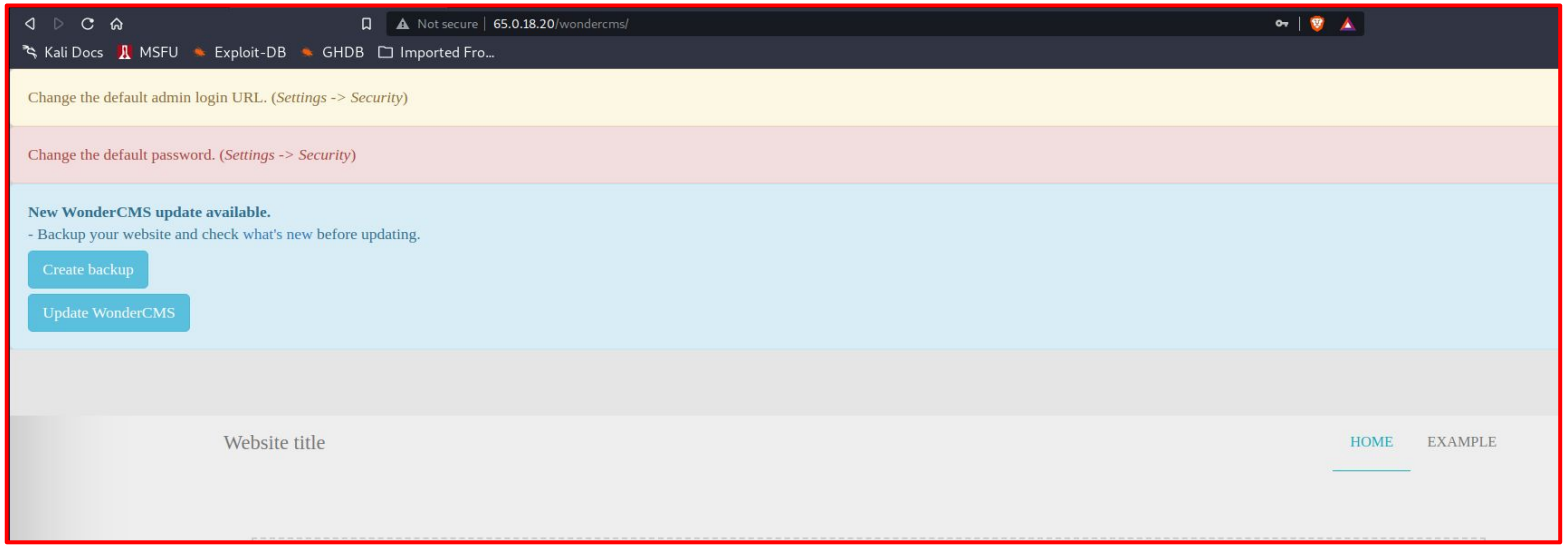
2) Access to Admin Panel (Critical)

Below mentioned URL is vulnerable to Default Password of Admin.

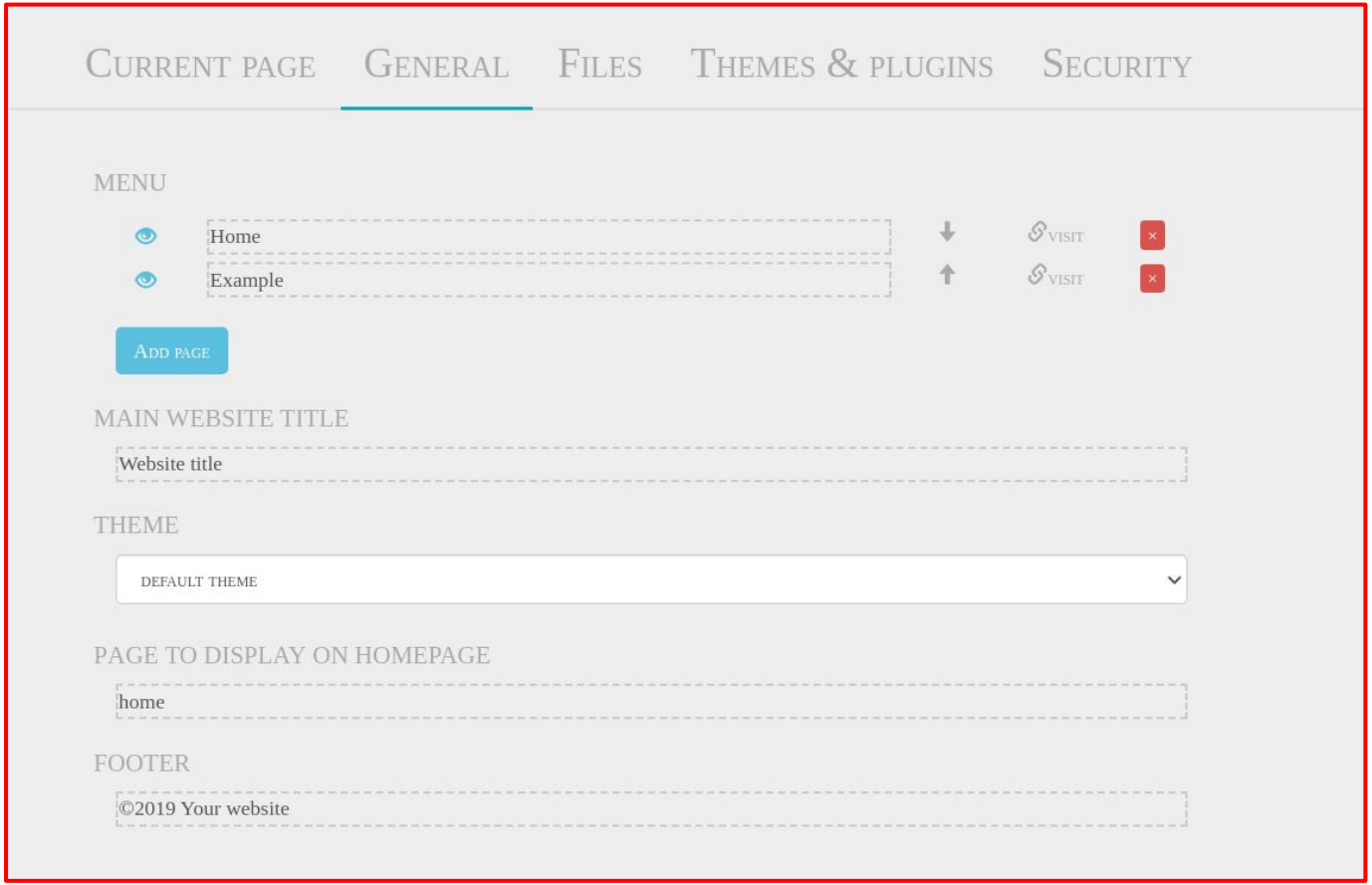
Affected URL :
•http://65.0.18.20/wondercms/loginURL

Observation

Hacker can logged into wonderCMS as admin with default password admin.
Hacker can also changed the password of admin so that next time admin don't get login.



Proof of Concept (POC)



Business Impact

1. Using this vulnerability, attacker can login to the admin panel and change the layout of the website. Attacker can also change the content of the website by this admin panel.
2. Attacker can add some malicious code, some kind of videos, blogs that are not belong to this website this is impact on company's reputation.
3. Attacker can add and delete the pages in this panel.
4. Attacker change the password of this admin panel so admin can not login in to this panel after attack.

Recommendation

1. The default password should be change into the strong password.
2. Password changing process must be done with some steps of verifications.
3. The admin url must also not accessible to normal user.

Reference

https://owasp.org/www-community/vulnerabilities/Use_of_hard-coded_password
https://www.acunetix.com/blog/web-security-zone/common-password-vulnerabilities/

4) Unauthorised access of customer details (IDOR)

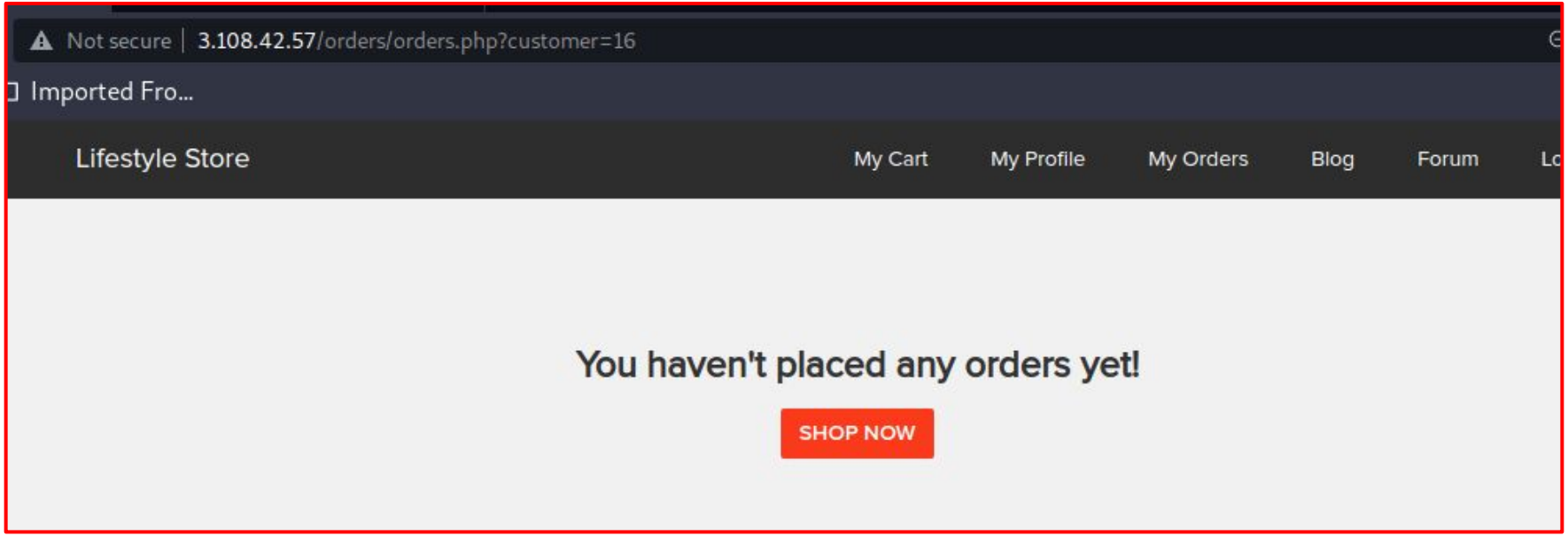
4. Unauthorised access of customer detail (Critical)

Below mentioned URL is vulnerable to Insecure Direct Object Reference.

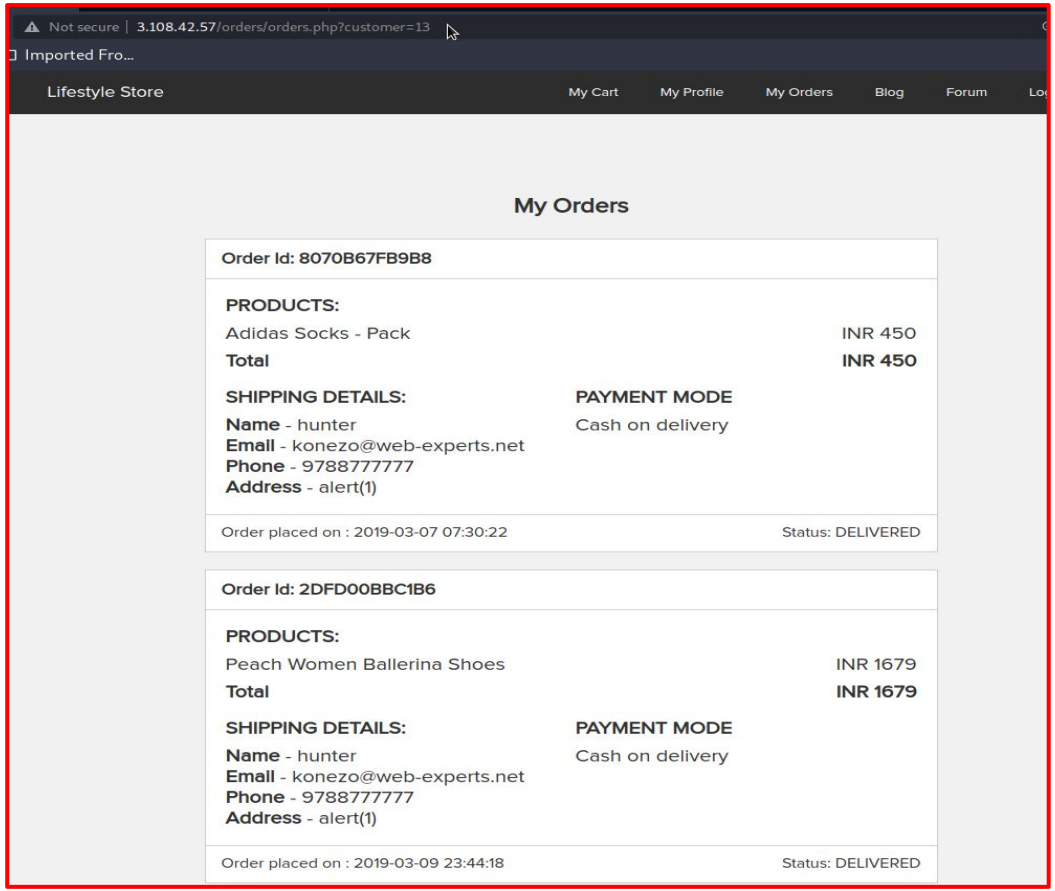
Affected URL :
•http://65.0.18.20/wondercms/loginURL

Observation

Login as customer then click on My Cart.
Change the customer parameter to any other number , we can see the item added by other customers into their cart.

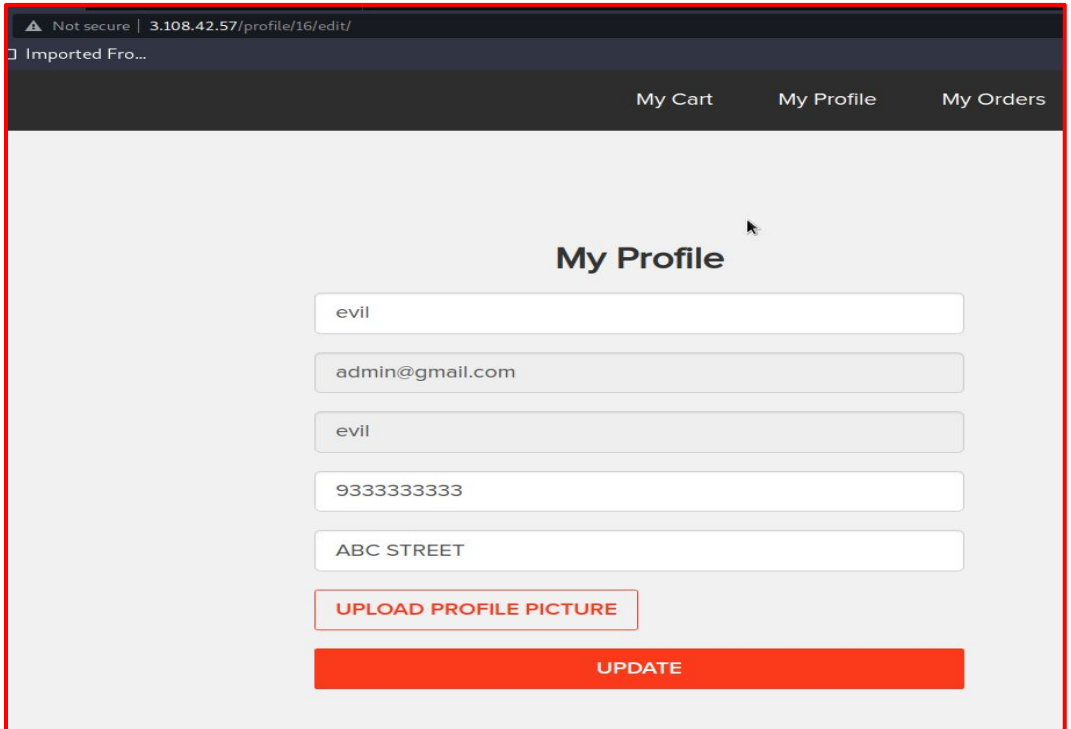


Proof of Concept (POC)

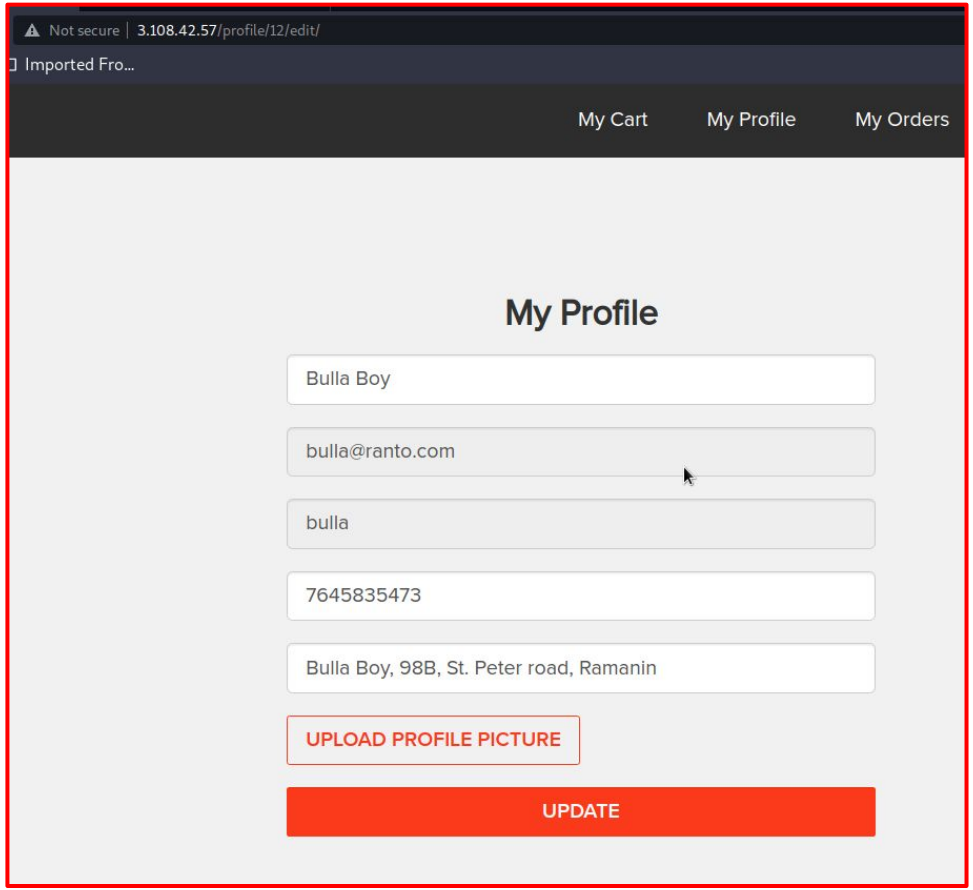


Observation

Login as a customer then in the url http://3.108.42.57/profile/16/edit/ when we change the number 16 to any other number we can see other users sensitive information like name , email & phone number.

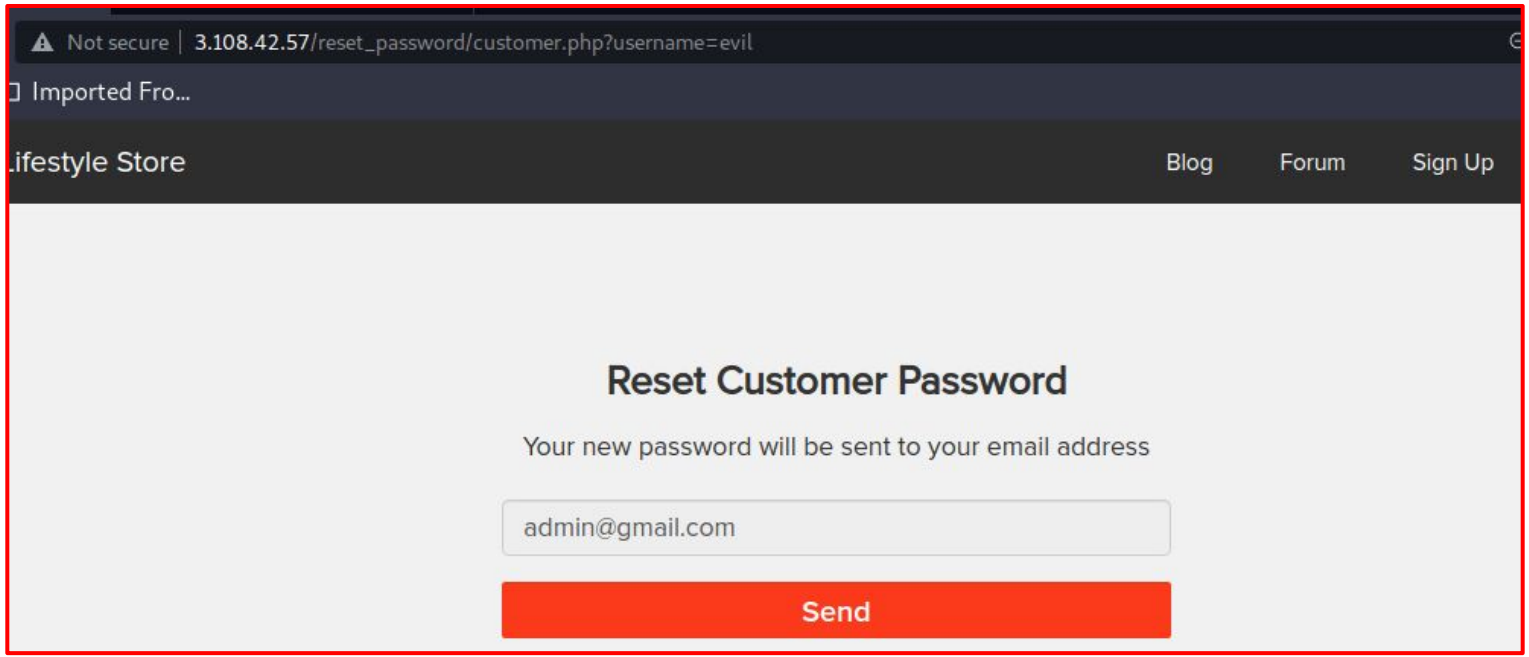


Proof of Concept (POC) : Sensitive information of other user

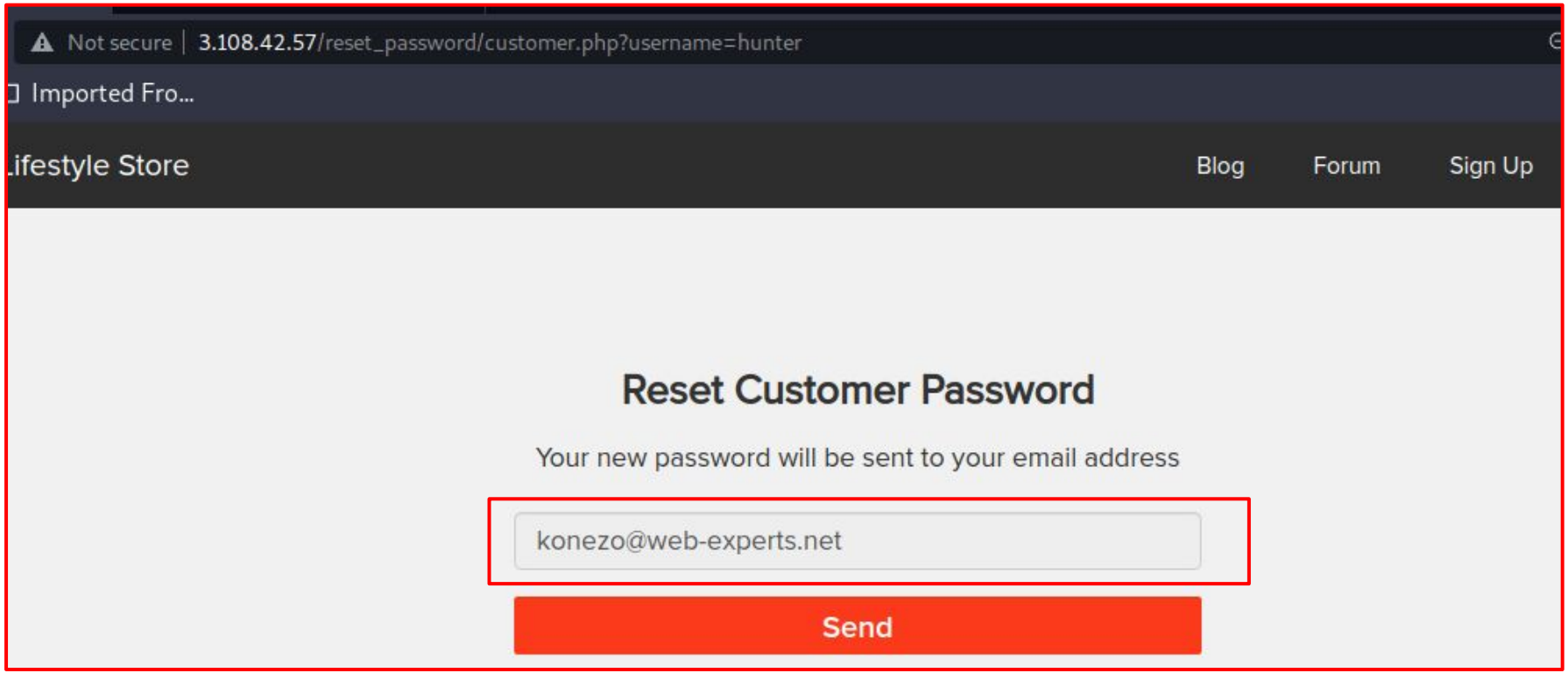


Observation

Navigate to http://3.108.42.57/reset_password/customer.php?username=xyz
Then we change the username parameter to any other username now an attacker can see the email id of that user .



Proof of Concept (POC)



Business Impact

1. This vulnerability exposes the information of all kind of user. Using this vulnerability, attacker can collect the data or information about the user and use those data in full fledge social engineering attack on user.
2. In this vulnerability information like phone number, username, email etc. is exposed.
3. Using this information the attacker can loggedin easily.

Recommendation

1. Sensitive information must only be accessible to authorised users.
2. Implement proper authentication and authorisation checks at every function to make sure the user requesting access to a resource whether to view or edit is his own data and no one else's.
3. Implement these checks on the basis of IP addresses and sessions.
4. If request can generate for reset password from different devices, the account should be blocked for a while.
5. Implement proper rate limiting checks that disallows large number of request from single resource.

Reference

1. https://hdivsecurity.com/bornsecure/insecure-direct-object-references-automatic-prevention/
2. https://gracefulsecurity.com/idor-insecure-direct-object-reference/

5) Reset password of admin by OTP Bypass

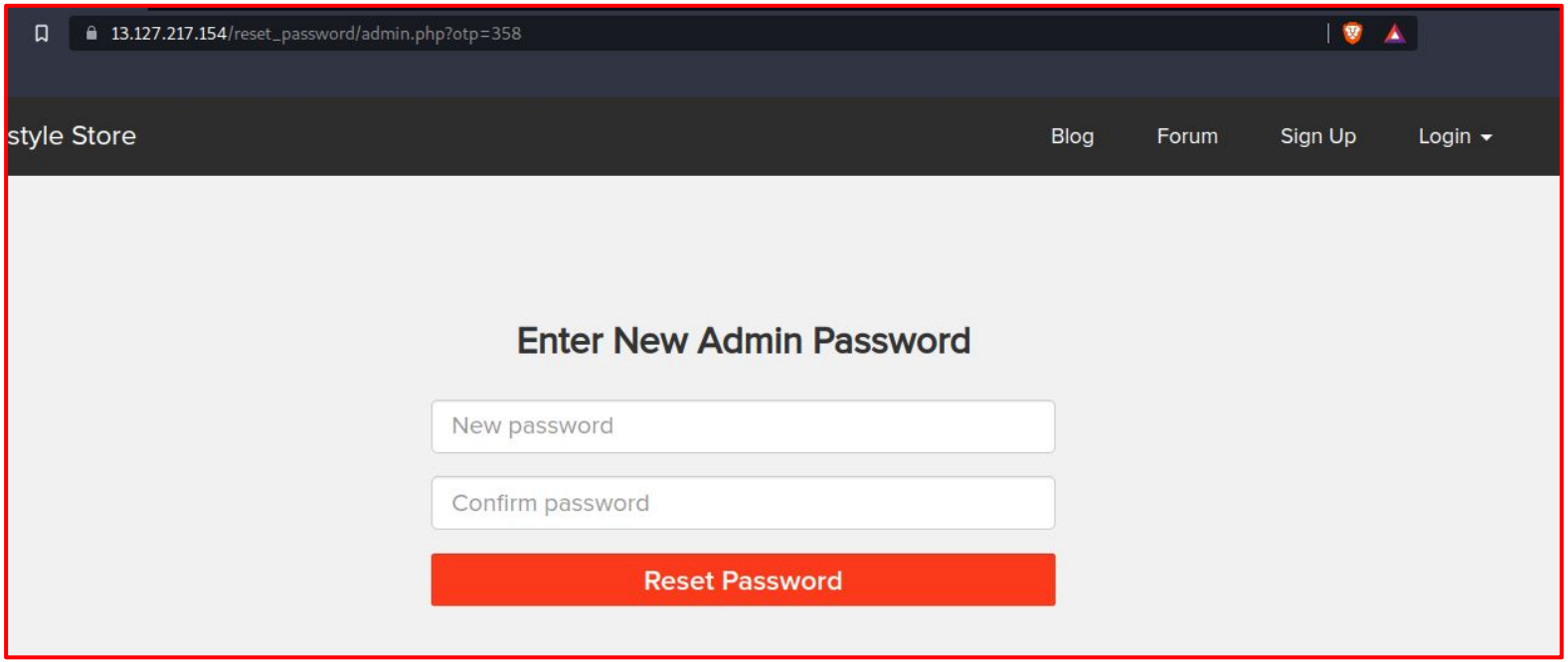
5. Reset password of admin by OTP Bypass (Critical)

Below mentioned URL is vulnerable to OTP Bypass Attack

Affected URL :
•http://url.com/reset_password/admin.php/otp=123

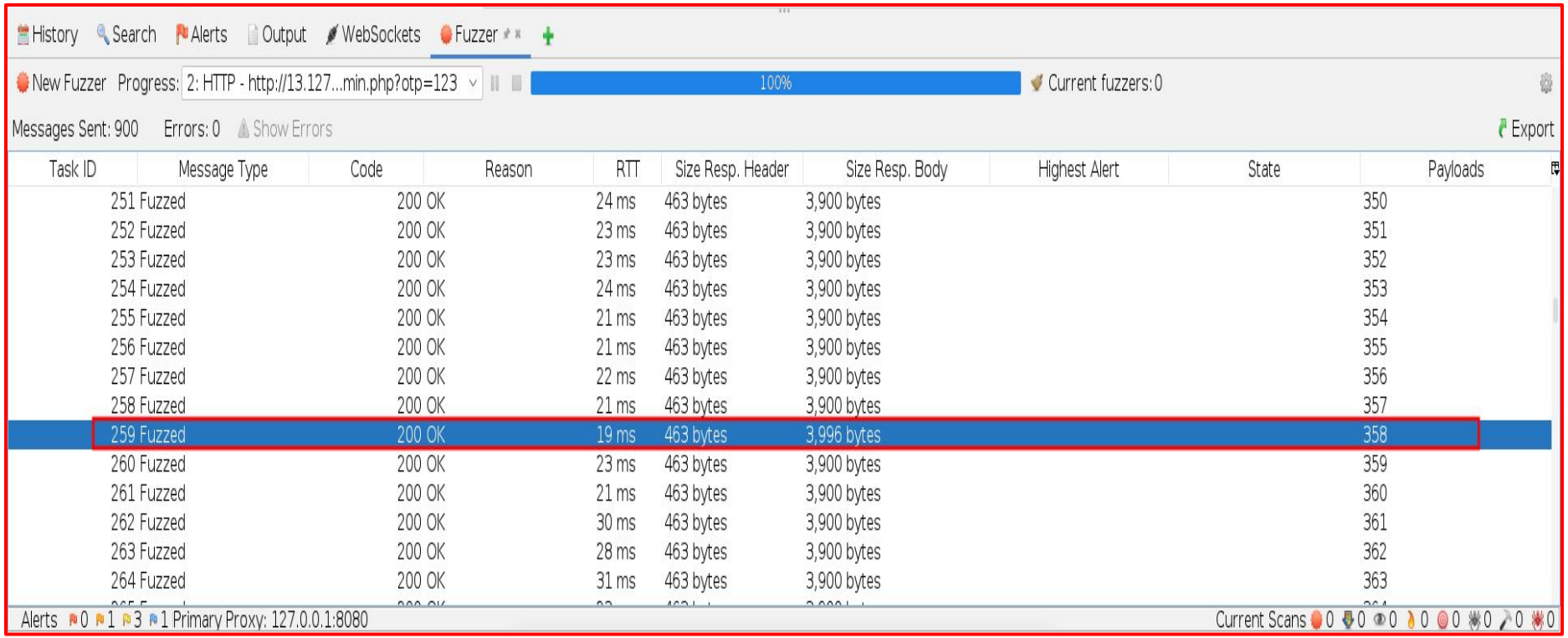
Observation

Admin login page option for forgot password and authenticate you by the OTP. The OTP parameter is GET based that means we will change the OTP in URL and check OTP is correct or not. An Attacker can easily brute force the 3 digit OTP.

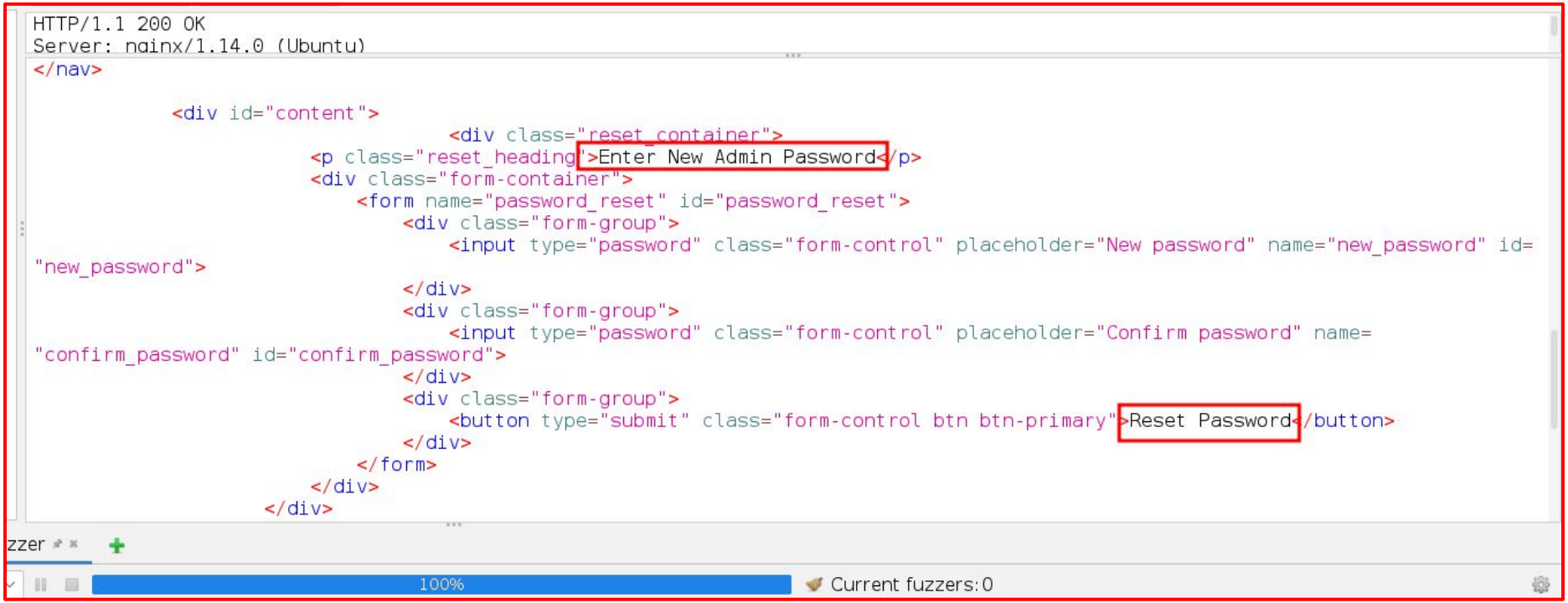


Proof of Concept (POC)

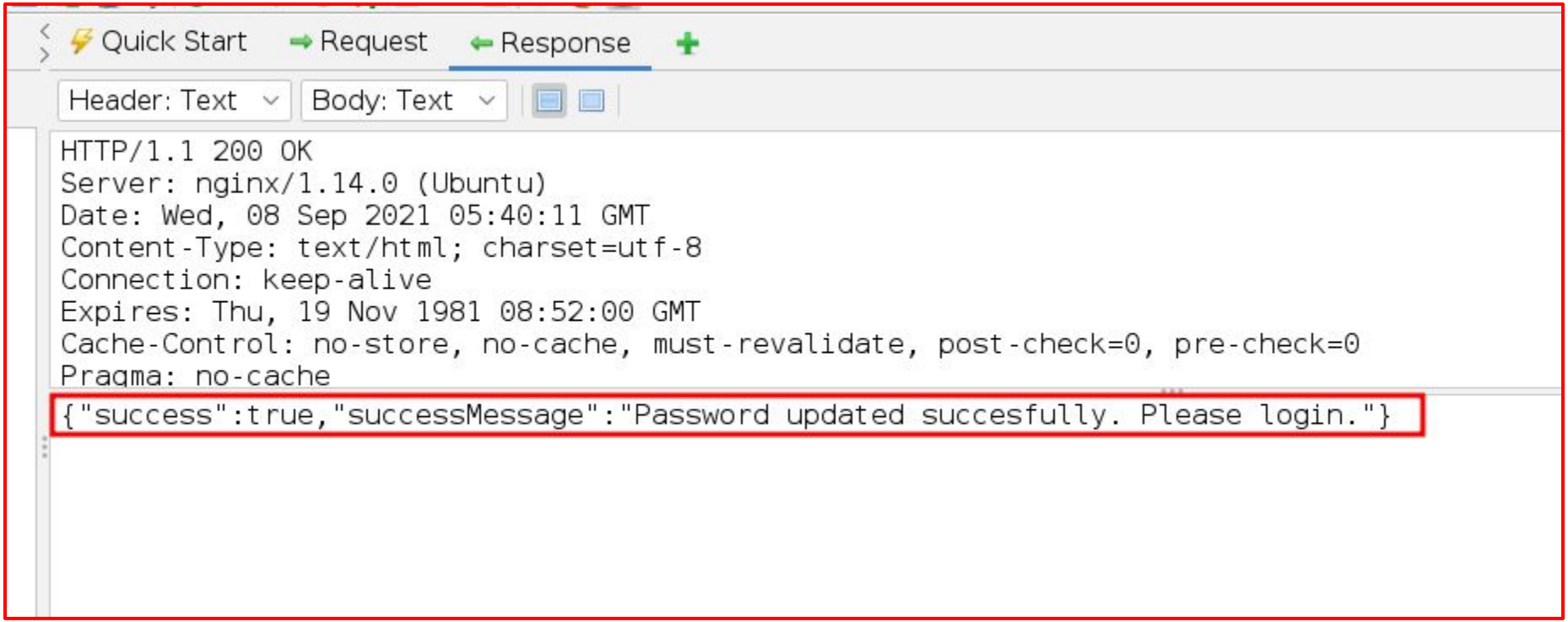
OTP is successfully brute forced.



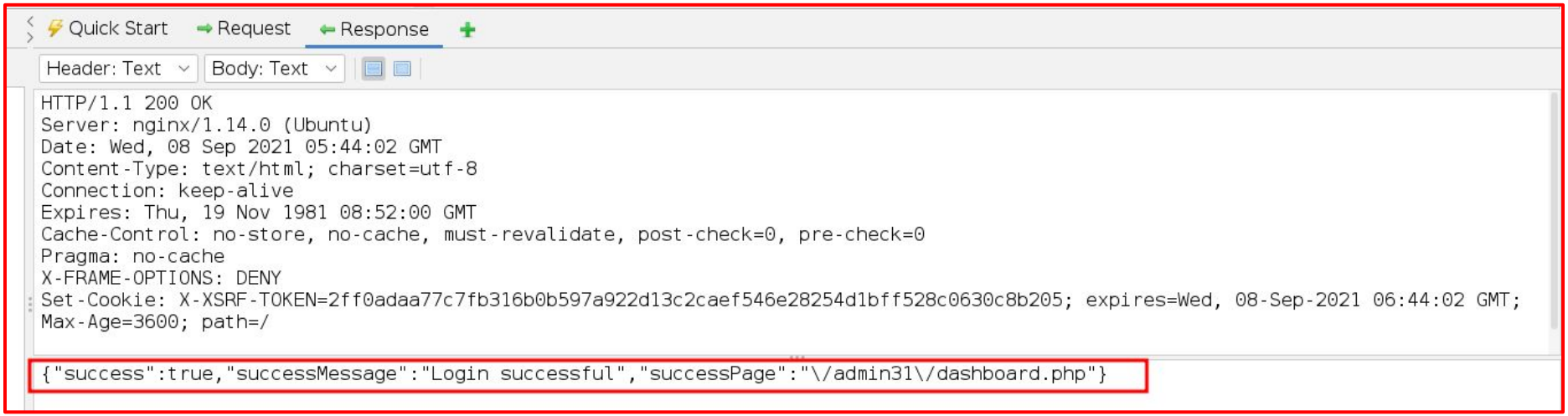
Proof of Concept (POC)



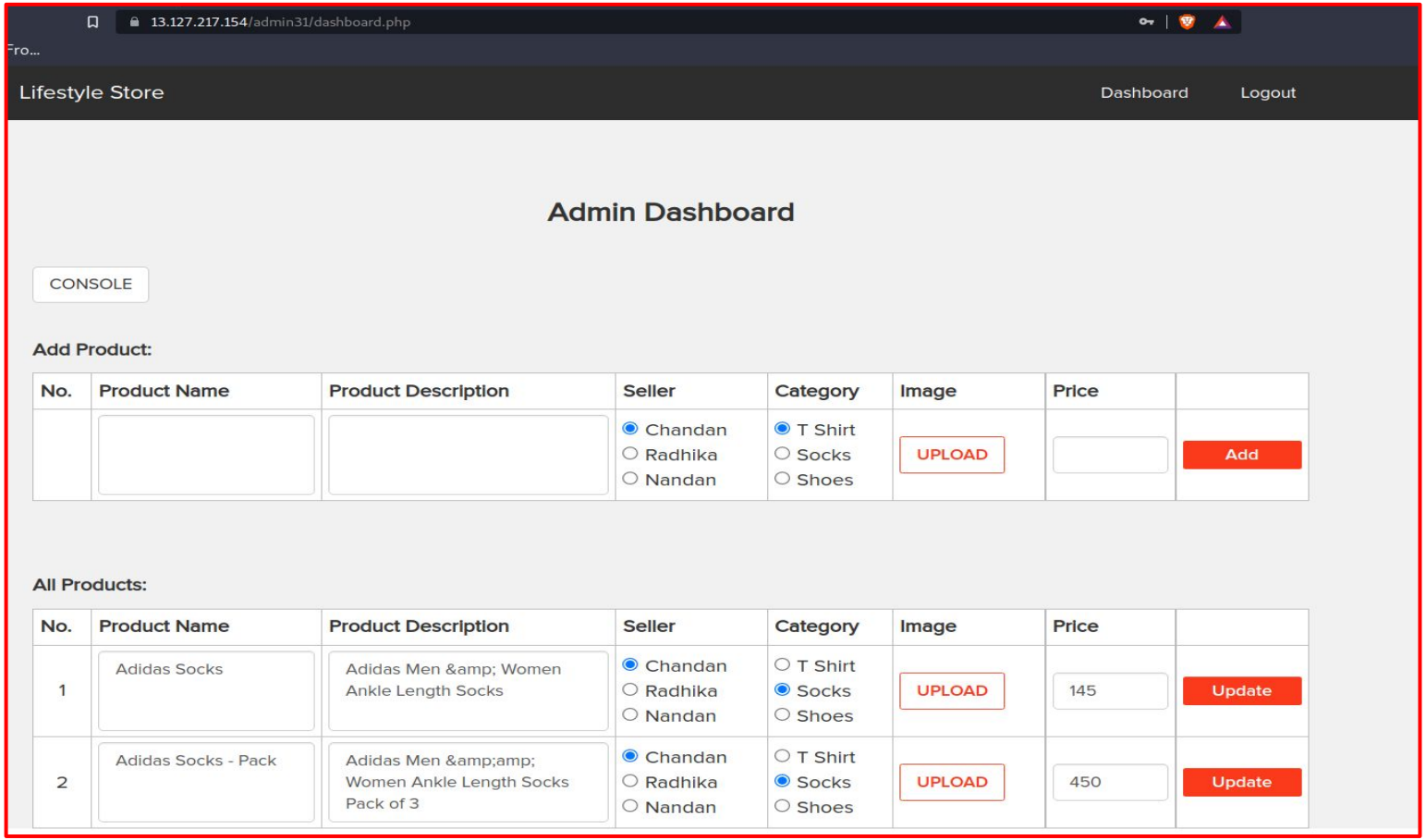
Proof of Concept (POC)



Proof of Concept (POC)



Proof of Concept (POC)



Business Impact

1. Using this vulnerability, attacker can use logical bruteforcing and steal the OTP and login into the admin account.
2. Attacker login in the admin account and full control of admin panel or website.
3. Attacker can add or delete the product and change the price of the product and so more.

Recommendation

1. Length of the OTP should be atleast 6. This makes bruteforcing hard.
2. Captcha can be used to protect from bruteforcing.
3. Number of attempts can be limited.
4. At least two-step verification before reset password.

Reference

1. https://owasp.org/www-community/attacks/Brute_force_attack
2. https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks
3. https://en.wikipedia.org/wiki/Brute-force_attack

6) Forced Browsing

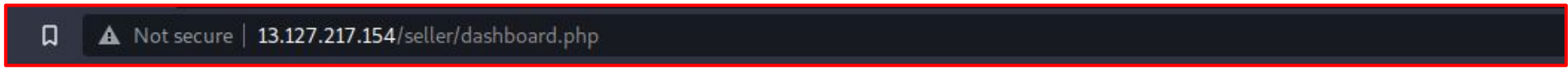
6. Forced Browsing (Critical)

Below mentioned URL is vulnerable to Forced Browsing.

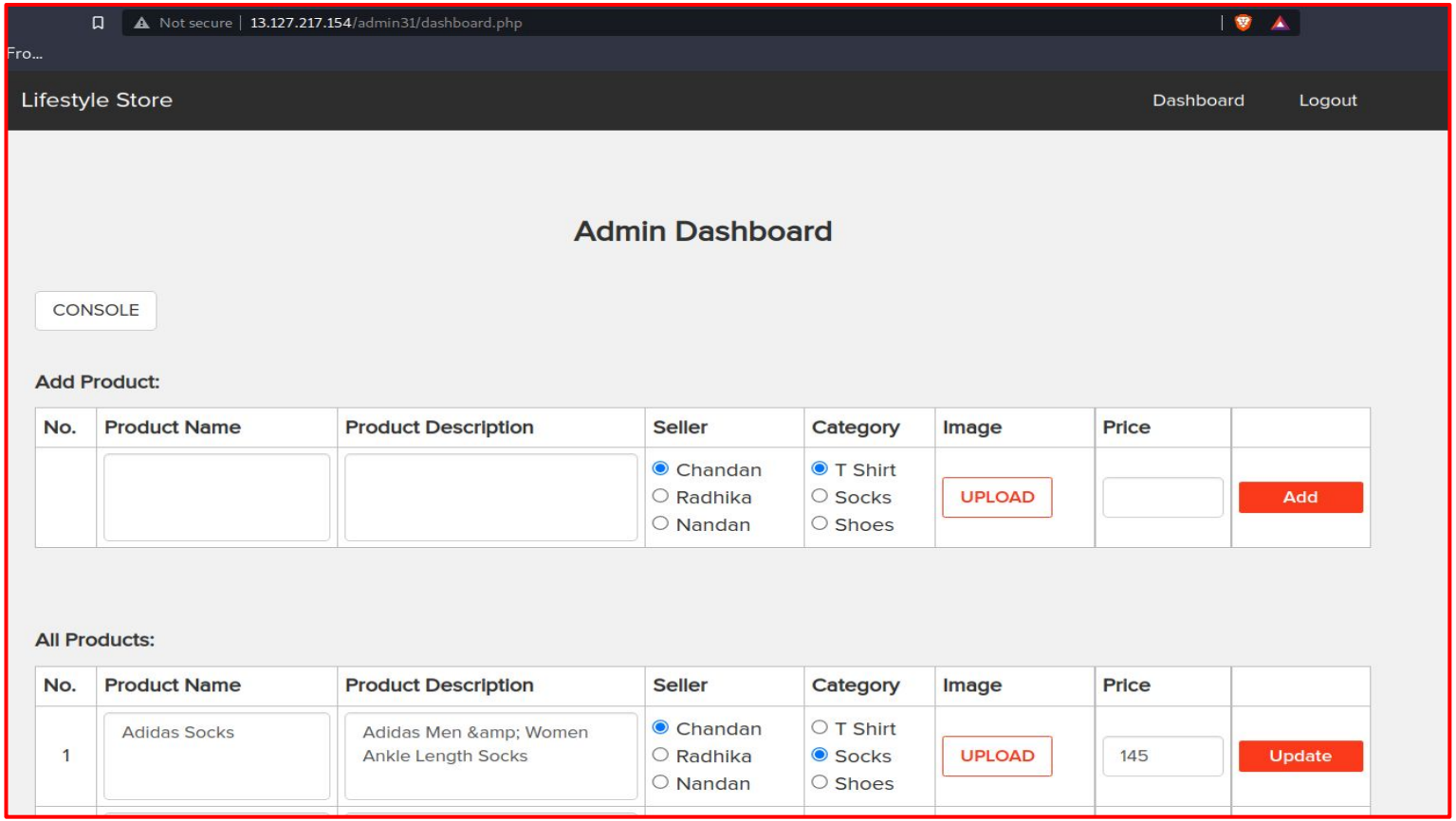
Affected URL :
•http://url.com/admin/dashboard.php

Observation

Login as a admin copy the admin URL
Then logout from admin account and login as a seller paste the admin URL onto Seller URL
admin dashboard will appear.



Proof of Concept (POC)



Business Impact

1. Using this vulnerability, seller can login into the admin panel after logout into the admin. Seller can paste the login url of admin panel in the seller's panel and he/she will be login in admin panel.
2. Attacker can change the price of the product and this cause financial loss.

Recommendation

1. After logout into the account can't access to login by login url
2. Using proper access control and authorization policies, access is only given to users commensurate with their privileges
3. Creating an allow list (or whitelist) involves allowing explicit access to a set of URLs that are considered to be a part of the application to exercise its functionality as intended. Any request not in this URL space is denied by default.

Reference

1. https://owasp.org/www-community/attacks/Forced_browsing
2. http://www.imperva.com/application_defense_center/glossary/forceful_browsing.html
3. https://campus.barracuda.com/product/webapplicationfirewall/doc/42049348/forced-browsing-attack/

7) Run command in admin panel

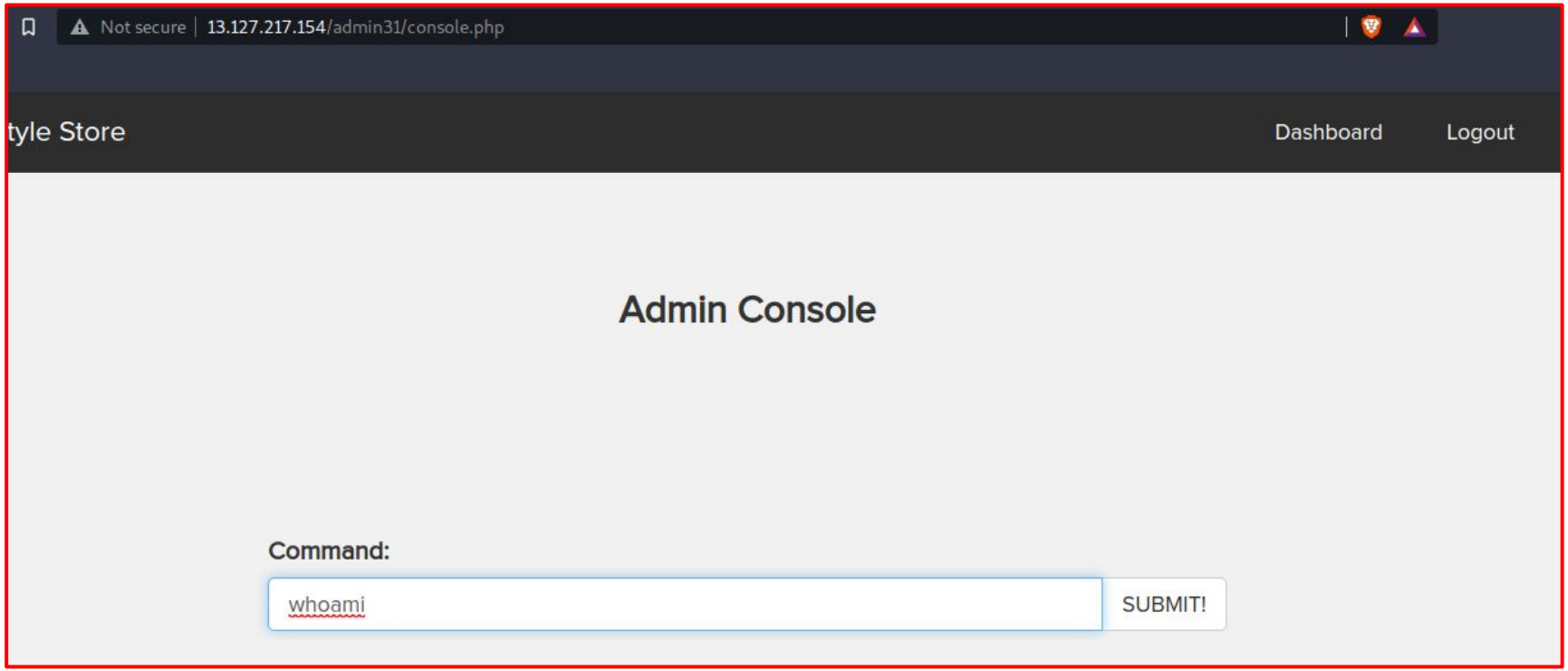
7. Run command in admin panel (Critical)

Below mentioned URL is vulnerable to command execution attack.

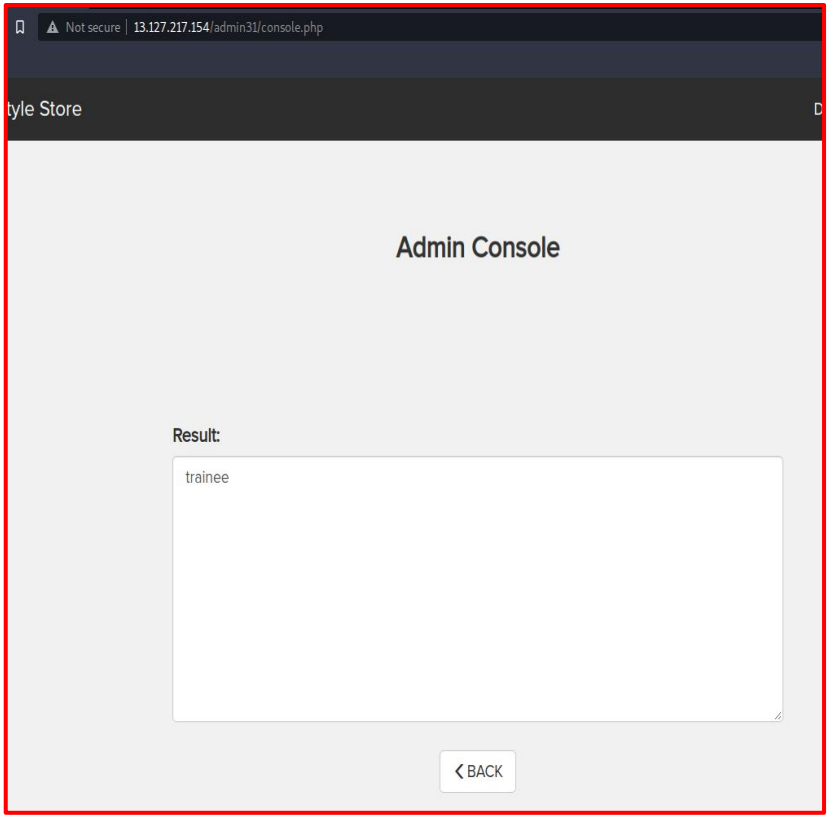
Affected URL :
•http://url.com/admin/console.php

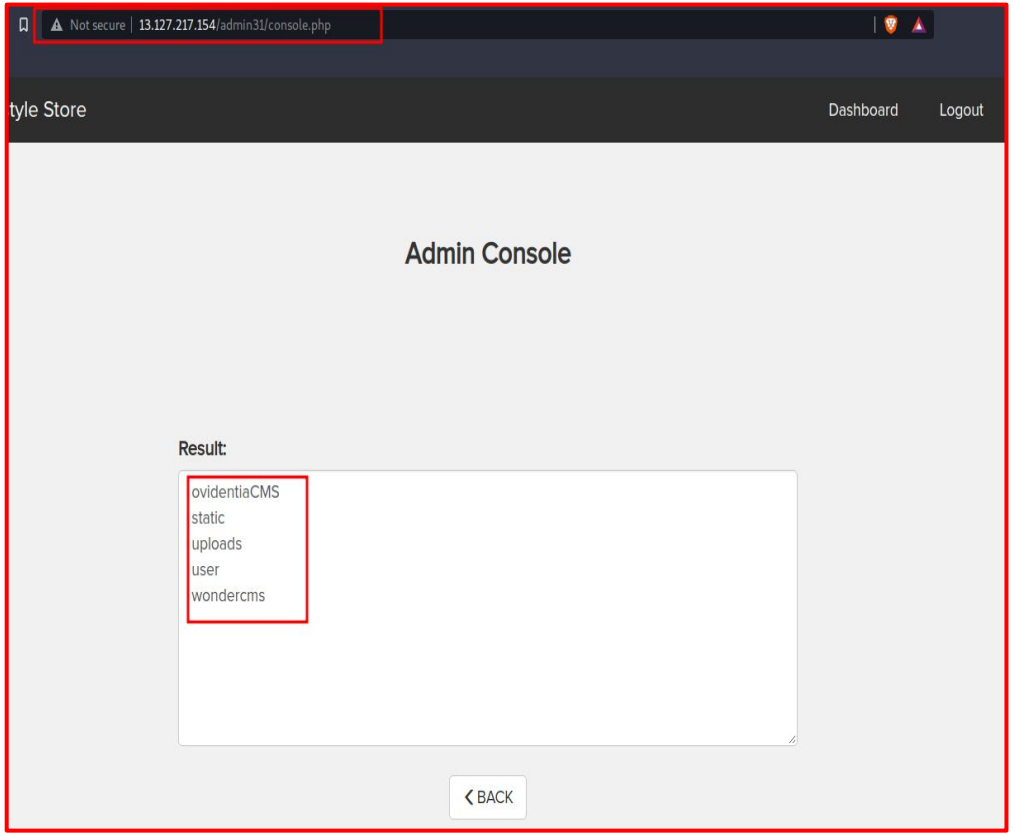
Observation

Login as admin then
Unix command can be executed and sensitive files can be seen by attacker.



Proof of Concept (POC)





Business Impact

1. Using this vulnerability, we can execute command in the admin panel and get result of command.
2. Attacker can add some malicious code or command to get more information about the website using this vulnerability.

Recommendation

1. There should be filters so that malicious code cannot be injected in .
2. Input validation can be done.
3. Output Validation can be done.
4. Canonicalization can also be done.

Reference

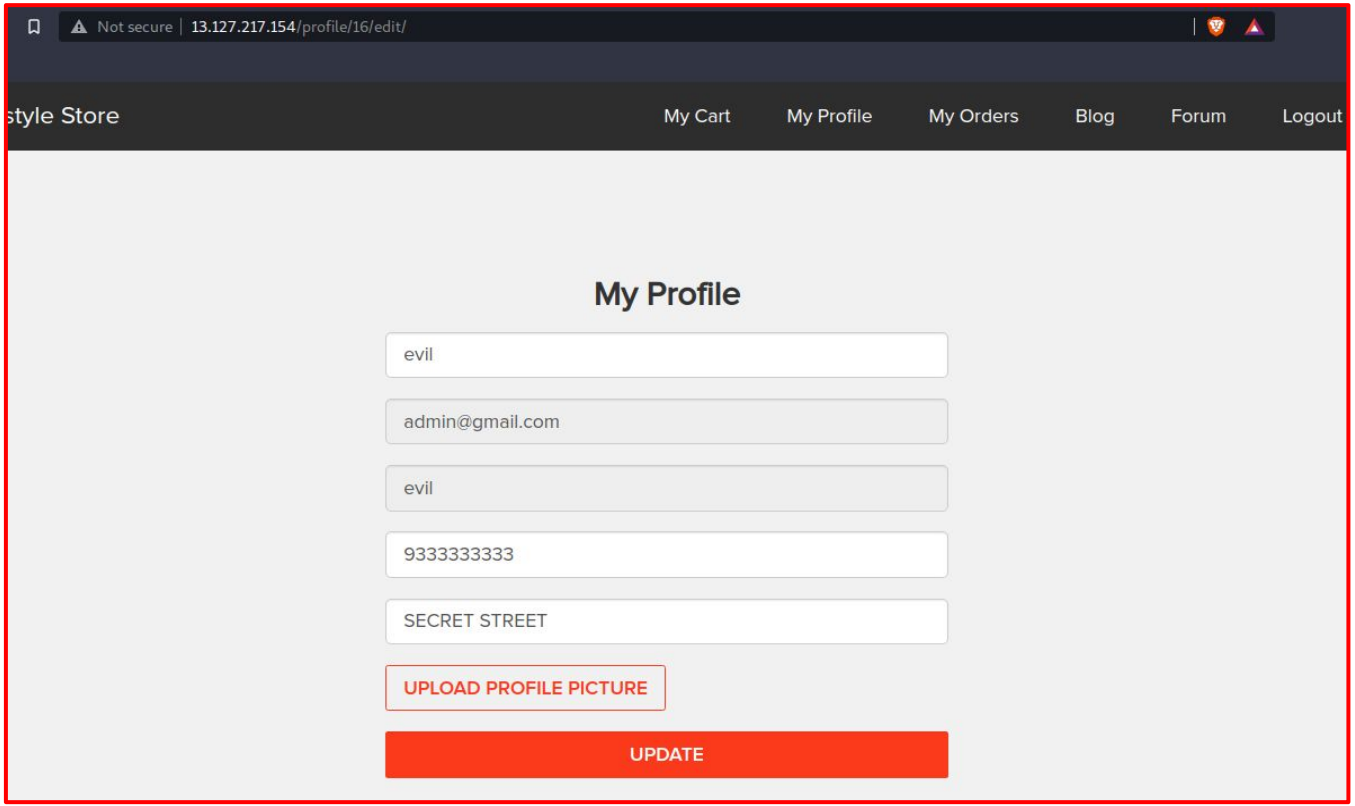
1. https://owasp.org/www-community/attacks/Command_Injection
2. https://cwe.mitre.org/data/definitions/77.html

8) Cross Site Request Forgery

8. Cross site request forgery (severe)	
Below mentioned URL is vulnerable to **CSRF** . **Affected URL :** http://url.com/profile/change_password.php **Affected Parameters :** Update button (POST parameter) We can change the password. **Affected URL :** http://url.com/cart/cart.php **Affected Parameters :** Confirm order option (POST parameter)	**Affected URL :** http://url.com/profile/16/edit **Affected Parameters :** Name, Phone, Address (POST parameter)

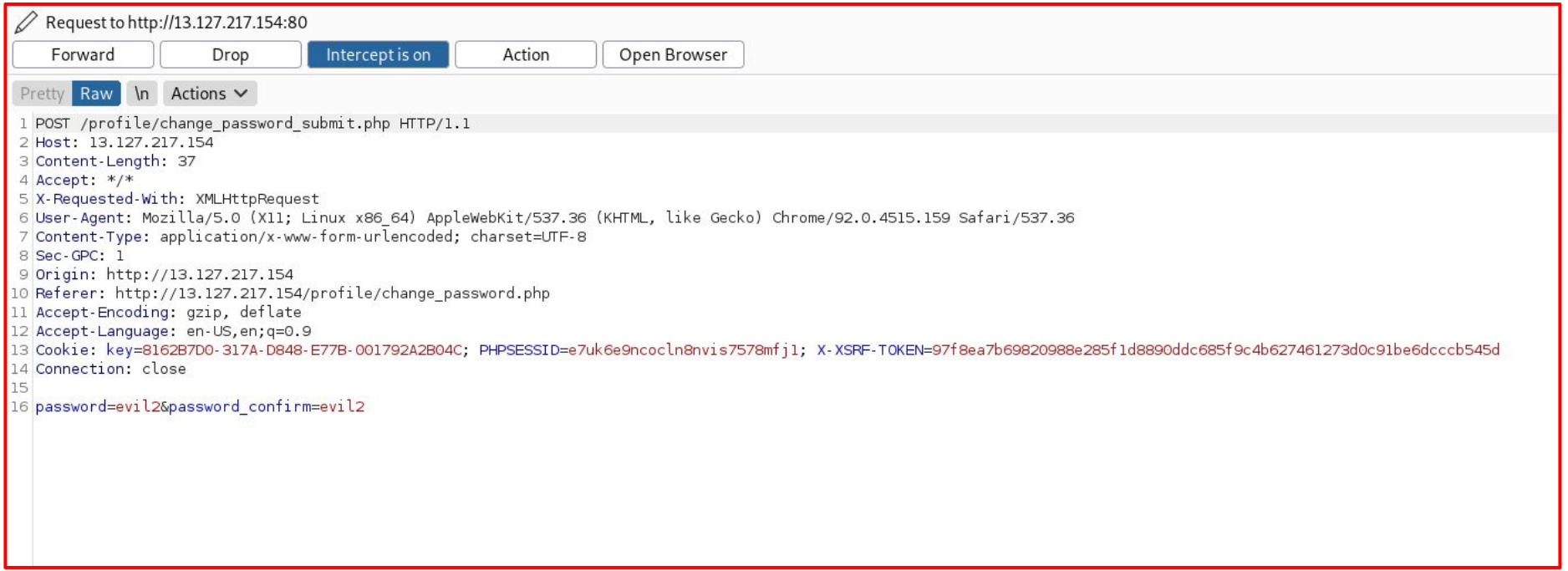
Observation

We navigate http://url.com/profile/16/edit after login into account.



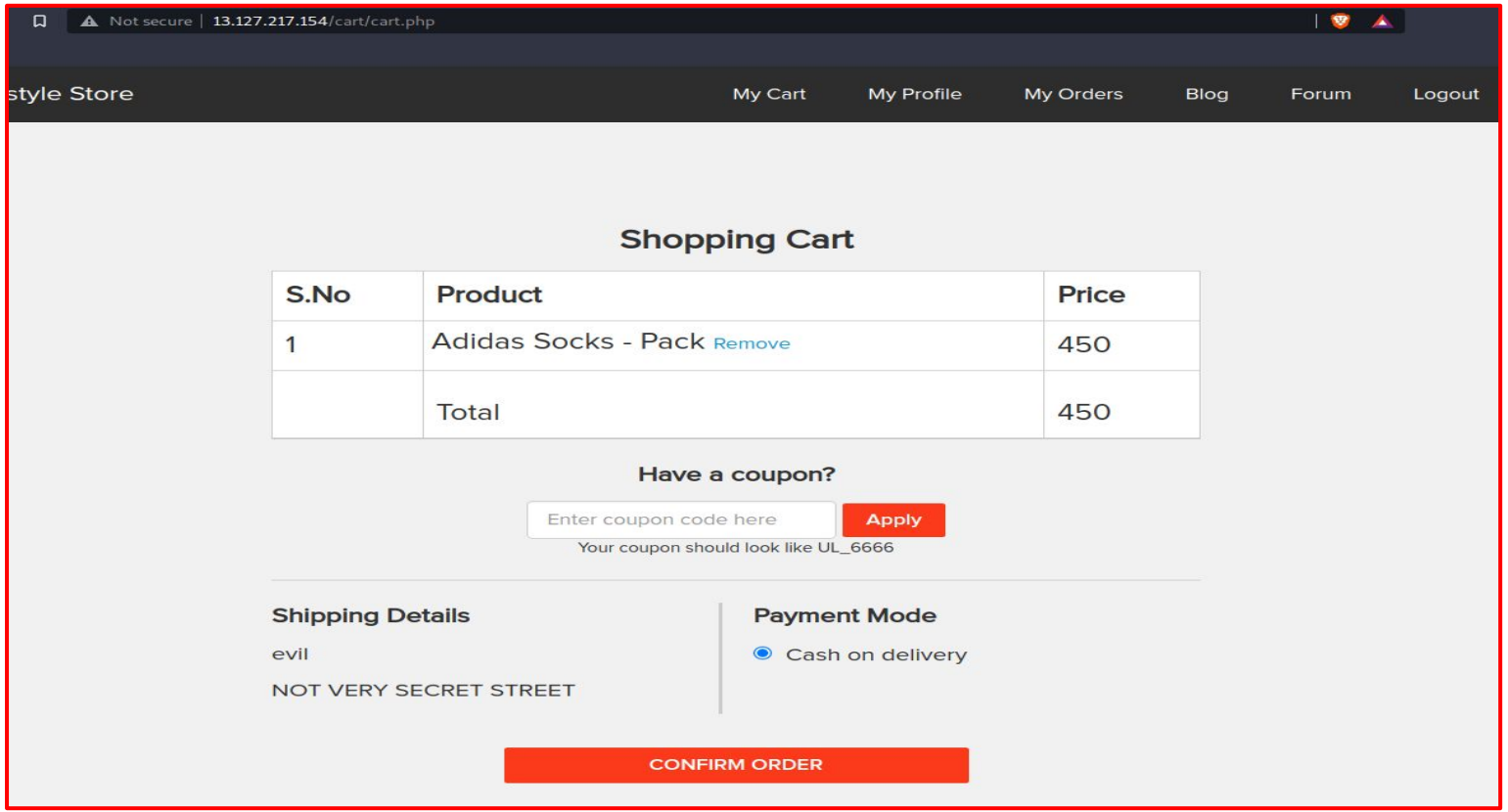
Observation

We navigate http://url.comprofile/change_password.php after login into account & capture the request into Burp Suite.



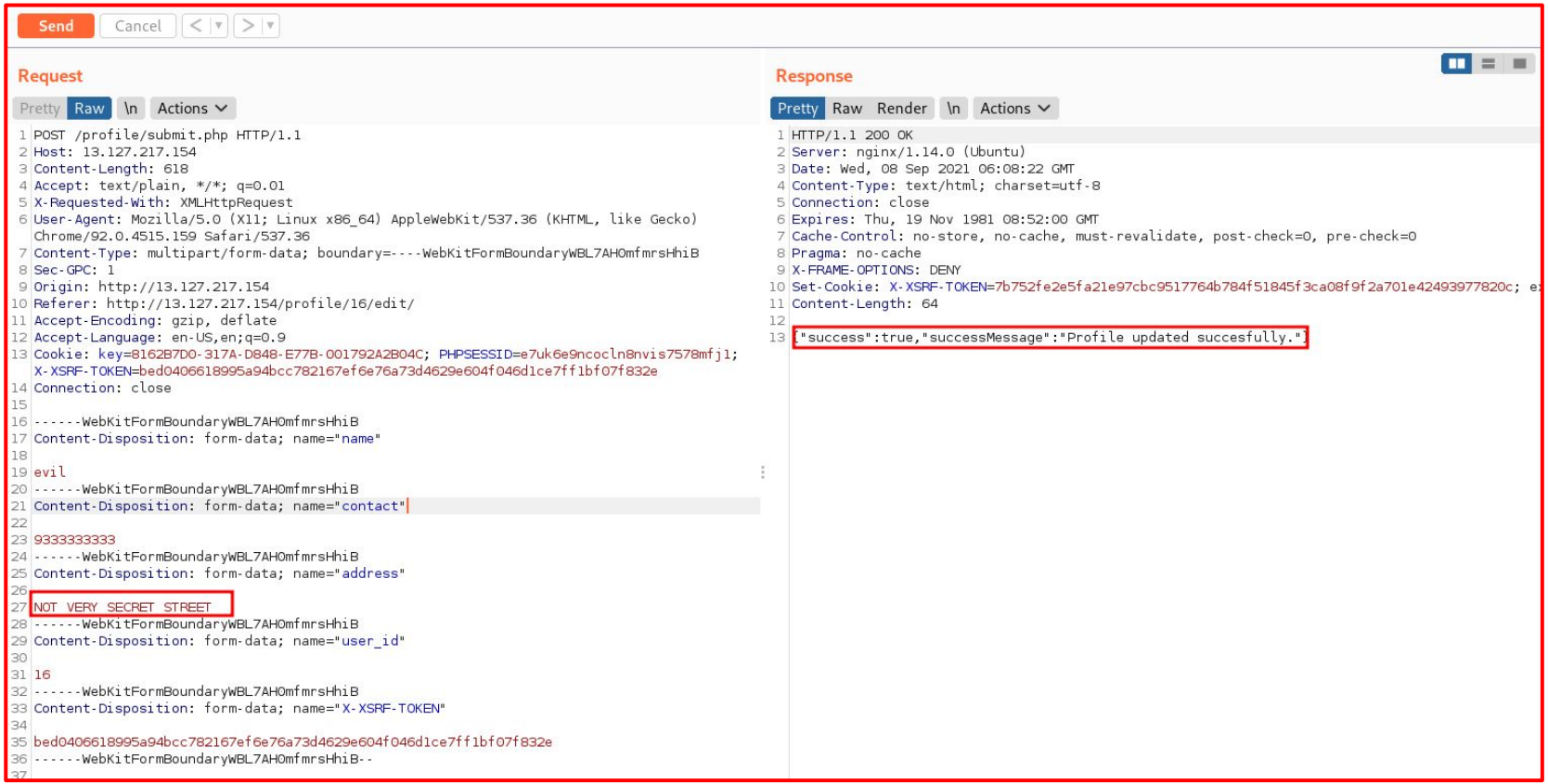
Observation

We select the product and add to cart for buy that product.



Proof of Concept (PoC)

We intercept the request of edit profile and tamper with the parameter. After tamper the data drop the original request and refresh the page.



Send Cancel

Request

Pretty Raw \n Actions

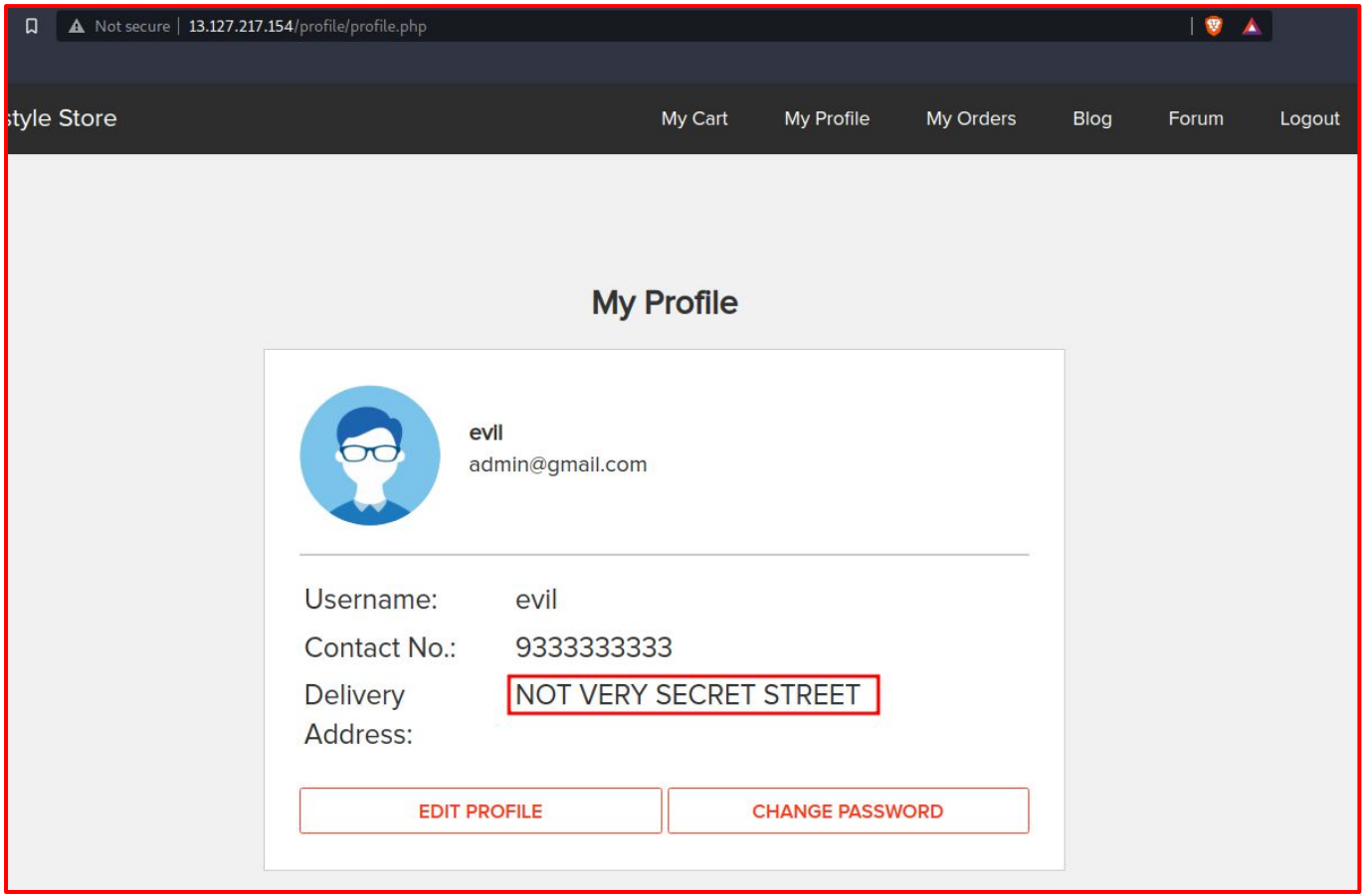
```
POST /profile/submit.php HTTP/1.1
Host: 13.127.217.154
Content-Length: 618
Accept: text/plain, */*; q=0.01
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/92.0.4515.159 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryWBL7AHOmfmrsHhiB
Sec-GPC: 1
Origin: http://13.127.217.154
Referer: http://13.127.217.154/profile/16/edit/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: key=8162B7D0-317A-D848-E77B-001792A2B04C; PHPSESSID=e7uk6e9ncocln8nvis7578mfj1;
X-XSRF-TOKEN=bed0406618995a94bcc782167ef6e76a73d4629e604f046d1ce7ff1bf07f832e
Connection: close

------WebKitFormBoundaryWBL7AHOmfmrsHhiB
Content-Disposition: form-data; name="name"

evil
------WebKitFormBoundaryWBL7AHOmfmrsHhiB
Content-Disposition: form-data; name="contact"

9333333333
------WebKitFormBoundaryWBL7AHOmfmrsHhiB
Content-Disposition: form-data; name="address"

NOT VERY SECRET STREET
------WebKitFormBoundaryWBL7AHOmfmrsHhiB
Content-Disposition: form-data; name="user_id"

16
------WebKitFormBoundaryWBL7AHOmfmrsHhiB
Content-Disposition: form-data; name="X-XSRF-TOKEN"

bed0406618995a94bcc782167ef6e76a73d4629e604f046d1ce7ff1bf07f832e
------WebKitFormBoundaryWBL7AHOmfmrsHhiB--
```

Response

Pretty Raw Render \n Actions

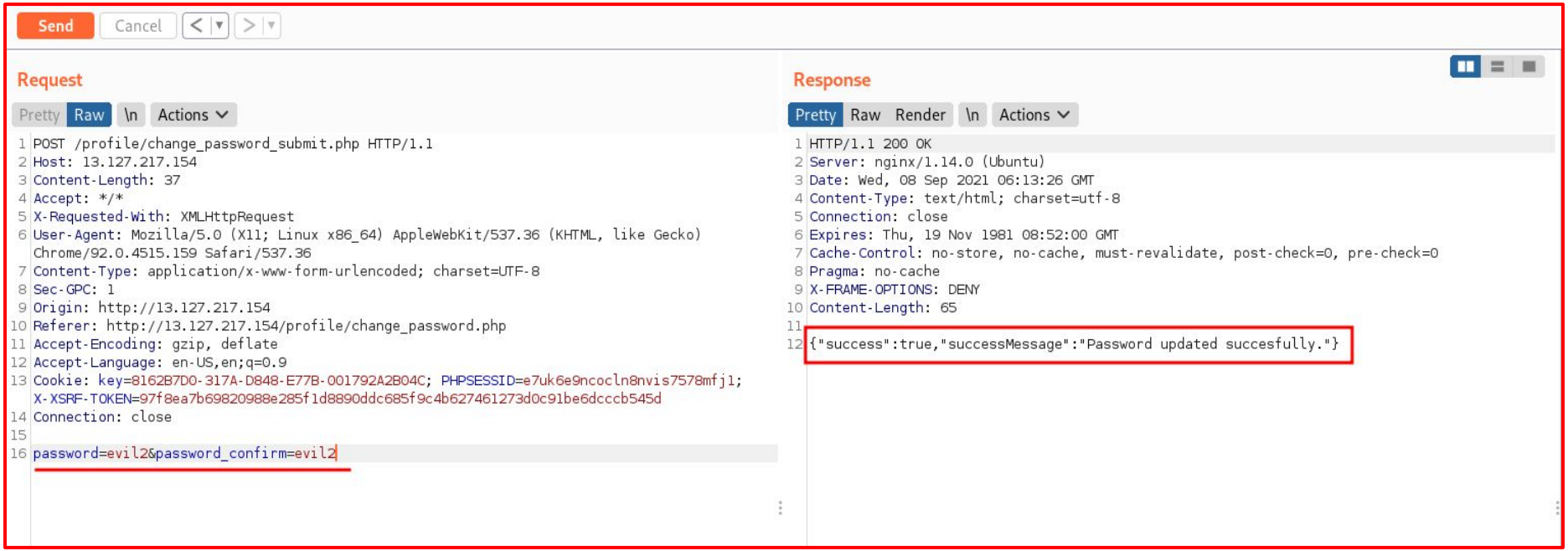
```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 08 Sep 2021 06:08:22 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-FRAME-OPTIONS: DENY
Set-Cookie: X-XSRF-TOKEN=7b752fe2e5fa21e97cbc9517764b784f51845f3ca08f9f2a701e42493977820c; e
Content-Length: 64

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

Proof of Concept (PoC)



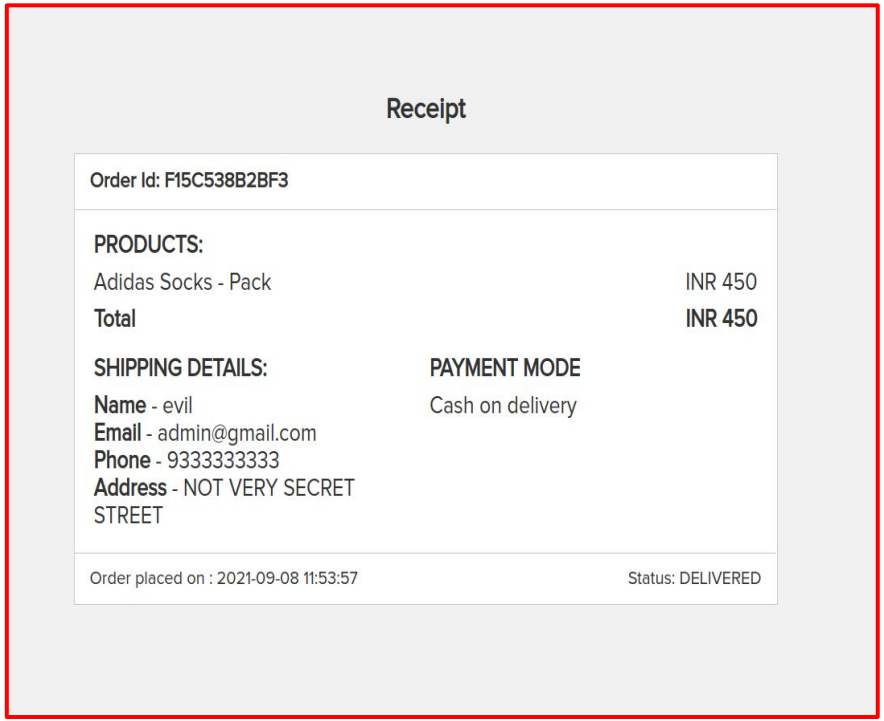
Proof of Concept (PoC)

We intercept the request of change password and tamper with the parameter password and confirm password. After tamper the data drop the original request and refresh the page.



Proof of Concept (PoC)

We create html code for confirm order and open in incognito window then refresh the page. After refresh the page order will be confirm.




```
<html>
<head>
</head>
<body>
<form action="http://13.127.217.154/orders/confirm.php" method="POST">
<input type="submit" value="Submit">  // CSRF Vulnerability
</form>
</body>
</html>
```

Business Impact

1. Using this vulnerability, attacker can able order many item which item user would cancel later when the sender will send it to him(as its cash on delivery),so unnecessary load of workers can increase incredibly.
2. Attacker can change the username, password, address, etc. with the help of this vulnerability.

Recommendation

1. Check the referer to before carrying out action.
2. 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.
3. 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

Reference

1. https://owasp.org/www-community/attacks/csrf
2. https://wiki.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005)

9) Default/Weak password

9. Default / Weak Password (severe)

Below mentioned URL is vulnerable to Default/Weak Password attack .

Affected URL :
•http://url.com/login/seller.php

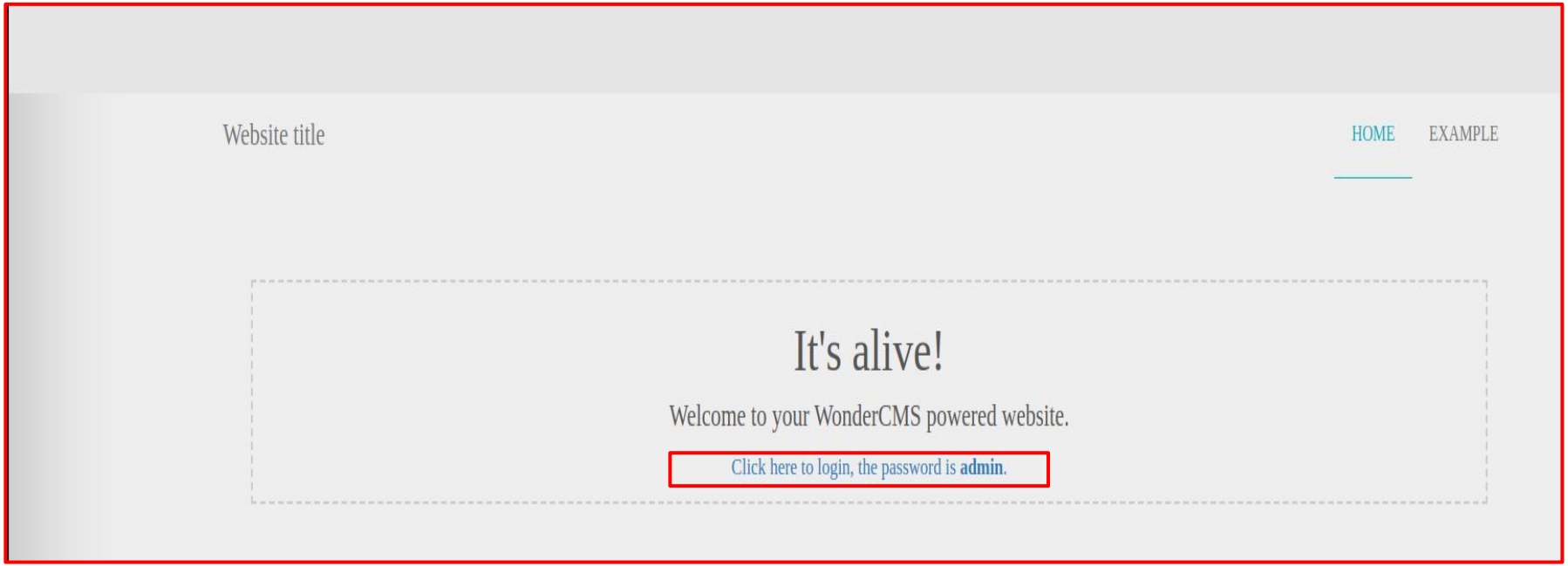
Affected Parameter :
Password (POST parameter)

Affected URL :
http://url.com/wondercms

Affected Parameter :
Password (POST parameter)

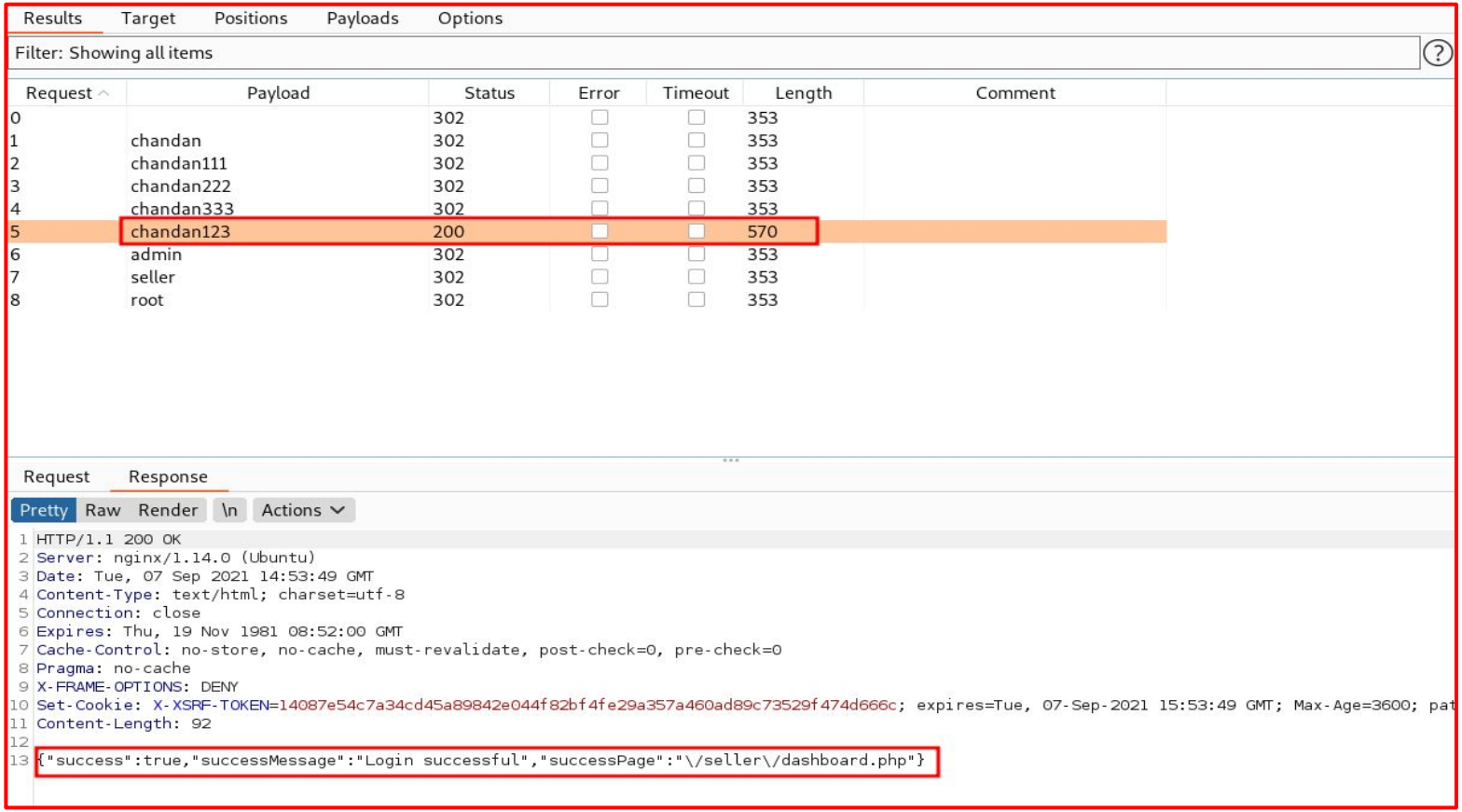
Observation

In the seller login panel seller has set weak password that can be easily guess to login into seller's account.
Blog adminhas set weak and default password as a admin. Very easy to login into blog panel with the default password.



Proof of Concept(PoC)

To login into seller panel we bruteforce the username and password in intruder.



Proof of Concept(PoC)

Login into blog we enter the default password : admin and we are login in the blog panel.

Business Impact

1. Using weak/default password attacker can login easily as a seller.
2. After login in to seller account the attacker can add or delete some data and also change the cost of the product this cause the financial loss for seller.
3. Attacker change the password of seller so, seller can’t login.

Recommendation

1. Change the password into strong password.
2. Captcha can be use to login as a seller.
3. Length of the password must at least 8 with alphanumeric character.

Reference

1. https://www.sciencedirect.com/topics/computer-science/default-password
2. https://owasp.org/www-community/vulnerabilities/Use_of_hard-coded_password

10) Cross Site Scripting (XSS)

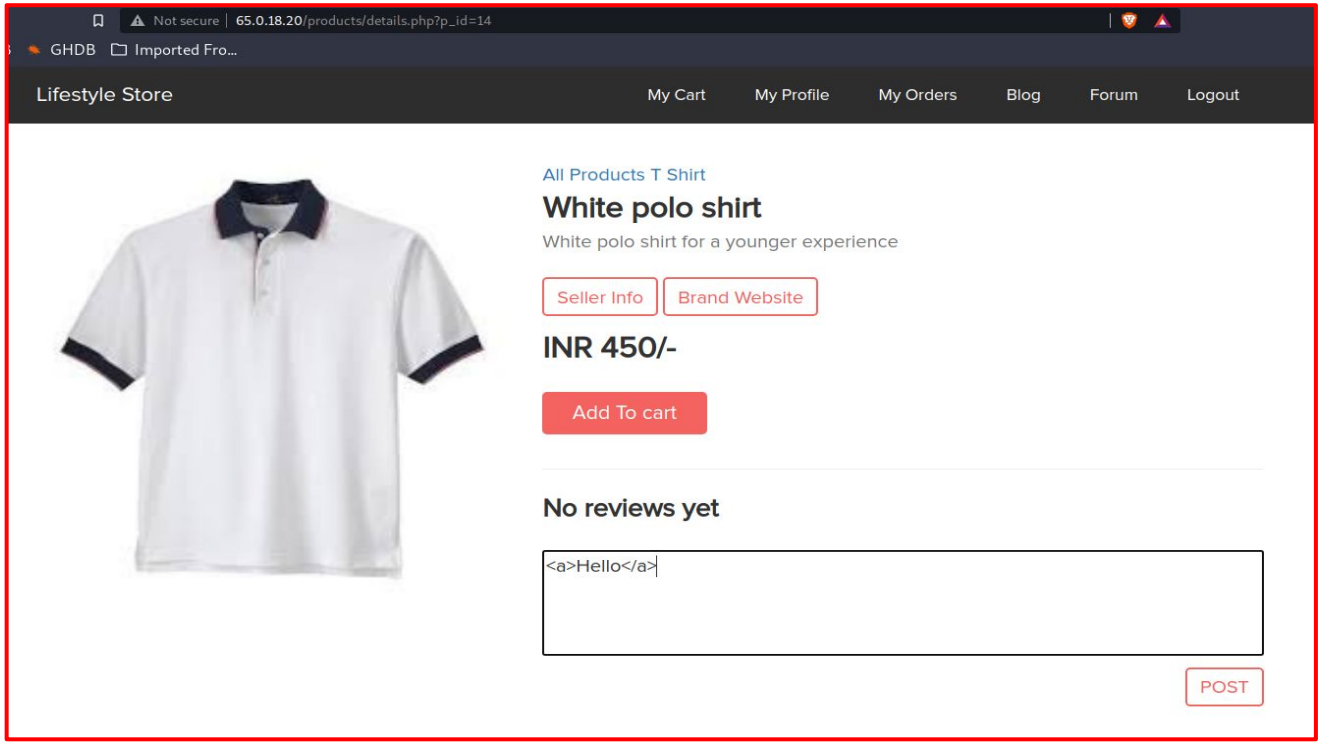
10. Cross site scripting (severe)	
Below mentioned URL is vulnerable to **XSS** . **Affected URL :** http://url.com/products/details.php/p_id=8 **Affected Parameters :** Review (POST parameter) **Affected URL :** http://url.com/wondercms **Affected Parameters :** Generals->main website title (POST parameter)	**Affected URL :** http://url.com/profile/16/edit **Affected Parameters :** Address (POST parameter)

Observation

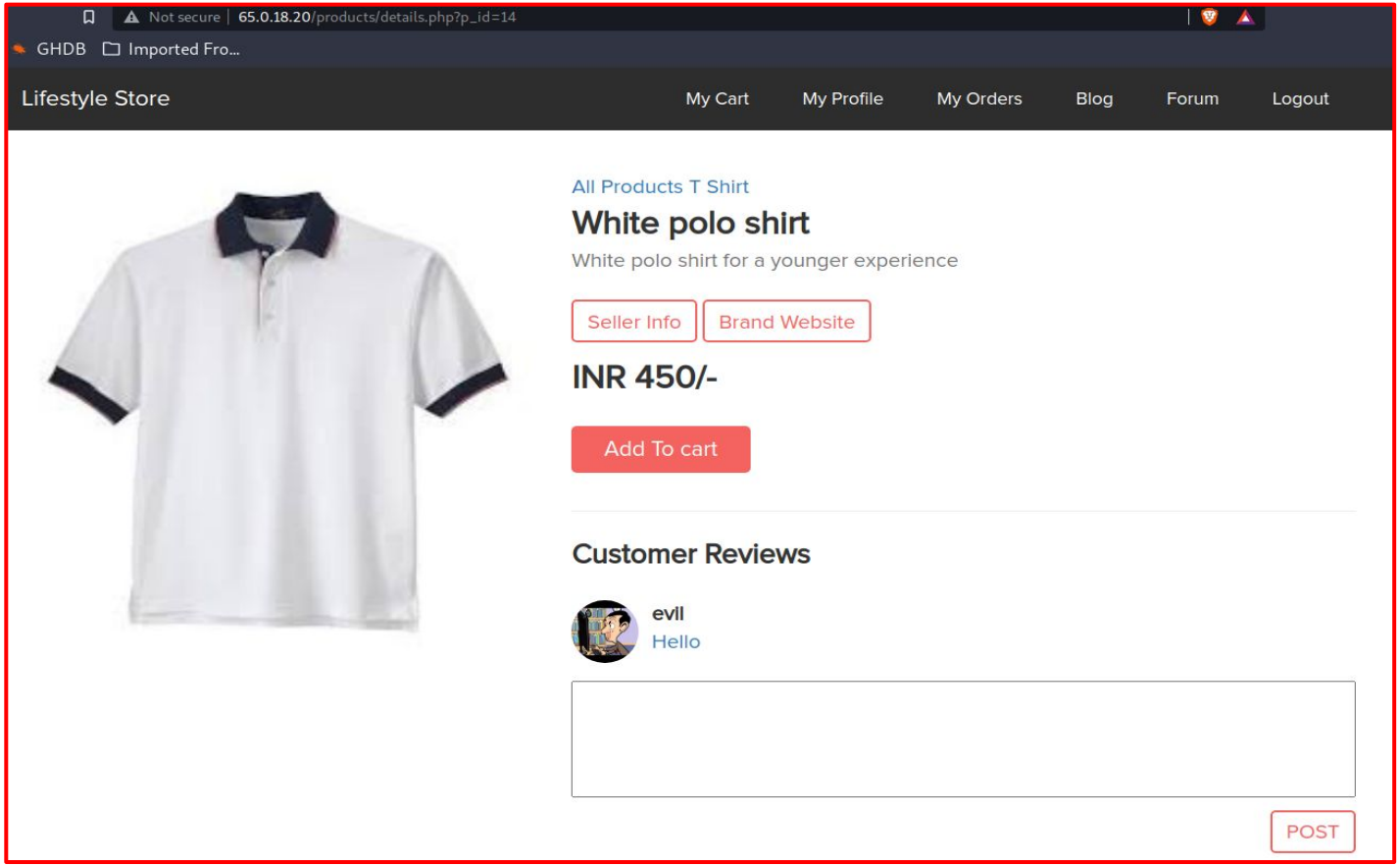
In the review box of product we pass the some parameter or write some html tags to check this is vulnerable or not.

This code is use to check this is vulnerable or not.

Code : <a> Hello



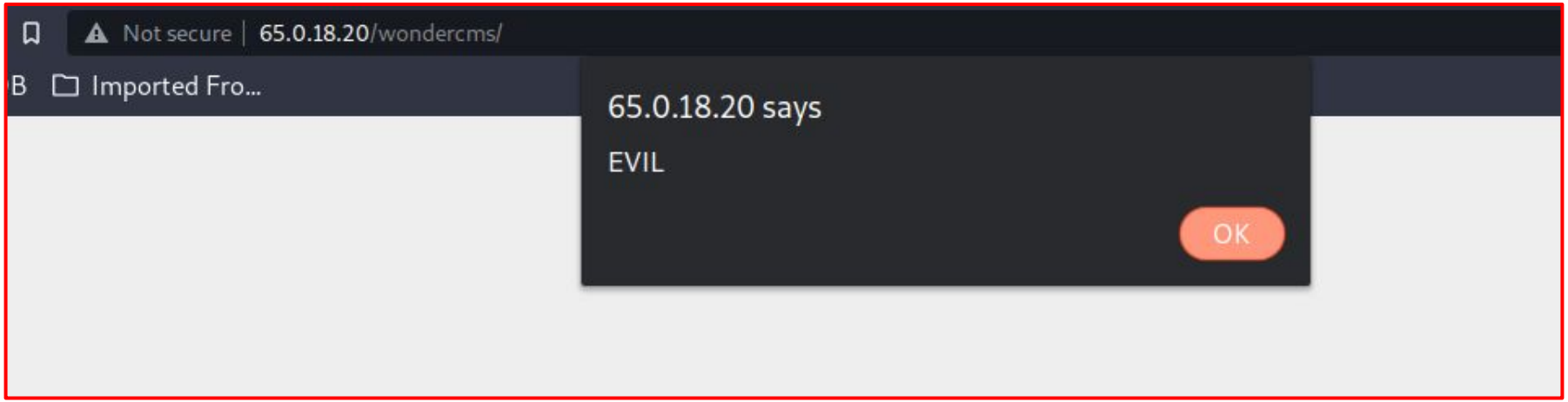
Proof of Concept (POC)



Observation

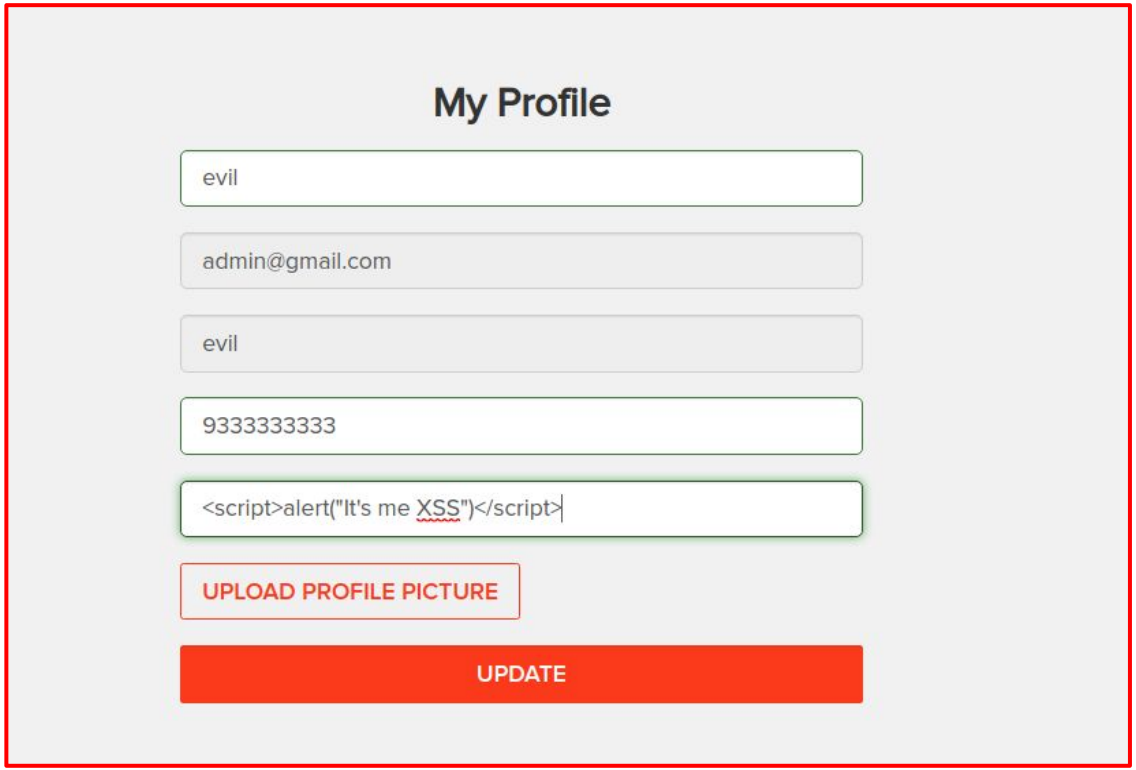
In the generals option we put special javascript code (like <script>alert(“EVIL”)</script>) for testing Cross Site Scripting. We get get same output at original page of website.

Proof of Concept (POC)

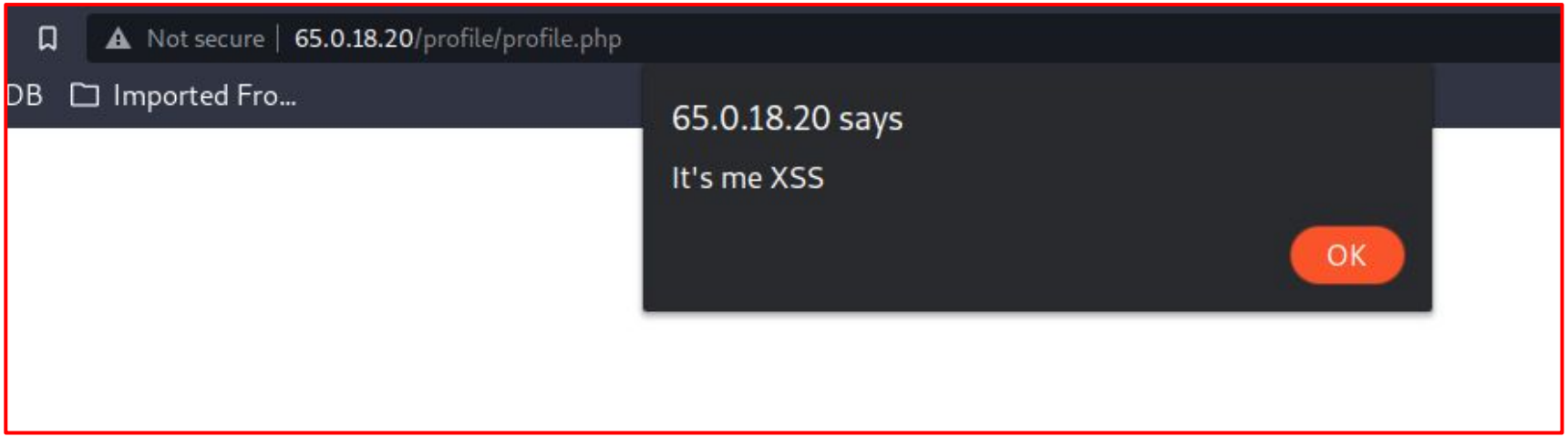


Observation

Go to the edit profile option after login. We put some special character in address field to check Cross site scripting vulnerability.



Proof of Concept (POC)



Business Impact

1. Using this vulnerability, attacker can inject some arbitrary like html, CSS, javascript via URL, attacker put any content on the page like phishing pages, install malware on victim's devices.
2. All attacker needs to do is send the link with the payload to the victim would see hacker controlled on the website. As the user trusts the website, he/she will trust the content

Recommendation

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

Reference

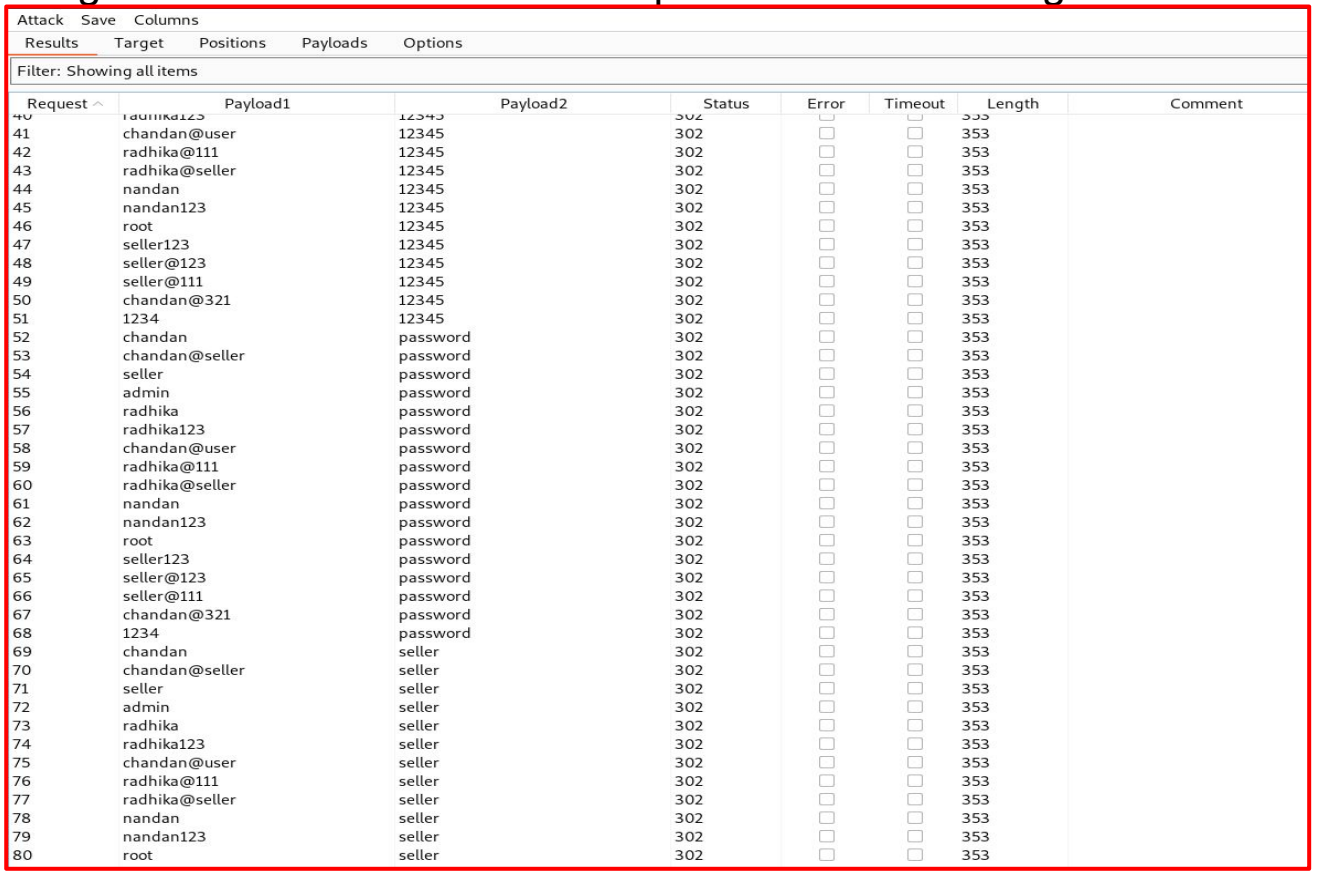
1. https://en.wikipedia.org/wiki/Cross-site_scripting
2. https://owasp.org/www-community/attacks/xss/
3. https://www.acunetix.com/websitesecurity/cross-site-scripting/

11) Rate Limiting Flaw

11. Rate Limiting Flaws (severe)	
Below mentioned URL is vulnerable to Rate Limiting Flaw . **Affected URL :** •http://url.com/login/seller.php **Affected Parameter :** Username, Password (POST parameter) **Affected URL :** http://url.com/login/customer.php **Affected Parameter :** Username, Password (POST parameter)	**Affected URL :** http://52.66.212.175/login/admin.php **Affected parameter :** Username,Password (POST parameter)

Observation

When put the credentials in the login fields we intercept this request in burpsuite, then send the request in intruder to change the value of username and password hence we get correct password and username.



Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
41	chandan@user	12345	302			353	
42	radhika@111	12345	302			353	
43	radhika@seller	12345	302			353	
44	nandan	12345	302			353	
45	nandan123	12345	302			353	
46	root	12345	302			353	
47	seller123	12345	302			353	
48	seller@123	12345	302			353	
49	seller@111	12345	302			353	
50	chandan@321	12345	302			353	
51	1234	12345	302			353	
52	chandan	password	302			353	
53	chandan@seller	password	302			353	
54	seller	password	302			353	
55	admin	password	302			353	
56	radhika	password	302			353	
57	radhika123	password	302			353	
58	chandan@user	password	302			353	
59	radhika@111	password	302			353	
60	radhika@seller	password	302			353	
61	nandan	password	302			353	
62	nandan123	password	302			353	
63	root	password	302			353	
64	seller123	password	302			353	
65	seller@123	password	302			353	
66	seller@111	password	302			353	
67	chandan@321	password	302			353	
68	1234	password	302			353	
69	chandan	seller	302			353	
70	chandan@seller	seller	302			353	
71	seller	seller	302			353	
72	admin	seller	302			353	
73	radhika	seller	302			353	
74	radhika123	seller	302			353	
75	chandan@user	seller	302			353	
76	radhika@111	seller	302			353	
77	radhika@seller	seller	302			353	
78	nandan	seller	302			353	
79	nandan123	seller	302			353	
80	root	seller	302			353	

Business Impact

1. Using this vulnerability, attacker can get the password using dictionary bruteforcing and easily get username and password of any login account.
2. Attacker can create lots of malicious account in this site by rate limiting flaws.

Recommendation

1. When the password are incorrect more than 5 times blocked that resource for some time.
2. Number of attempts can be limited.
3. The password length must be large so bruteforcing can be not possible.
4. Captcha should be used to protect from brute force.

Reference

1. https://medium.com/bugbountywriteup/bypassing-rate-limit-abusing-misconfiguration-rules-dcd38e4e1028
2. https://www.keycdn.com/support/rate-limiting
3. https://ussignal.com/blog/protect-against-cyber-attacks-with-rate-limiting

12) Open Redirection

12. Open Redirection (severe)

Below mentioned URL is vulnerable to Open Redirection attack.

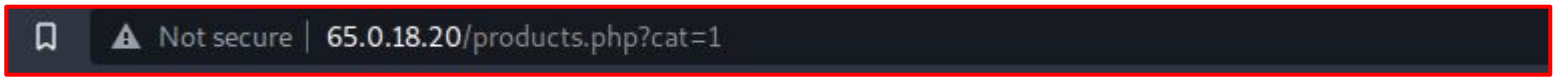
Affected URL :
http://url.com/products/cat=1

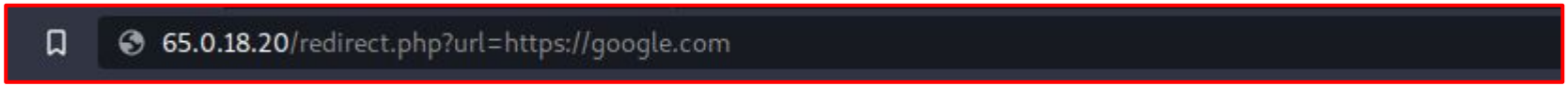
Affected Parameter :
url

Observation

Url of product is in GET based parameter so we can change the url and redirect to other site.

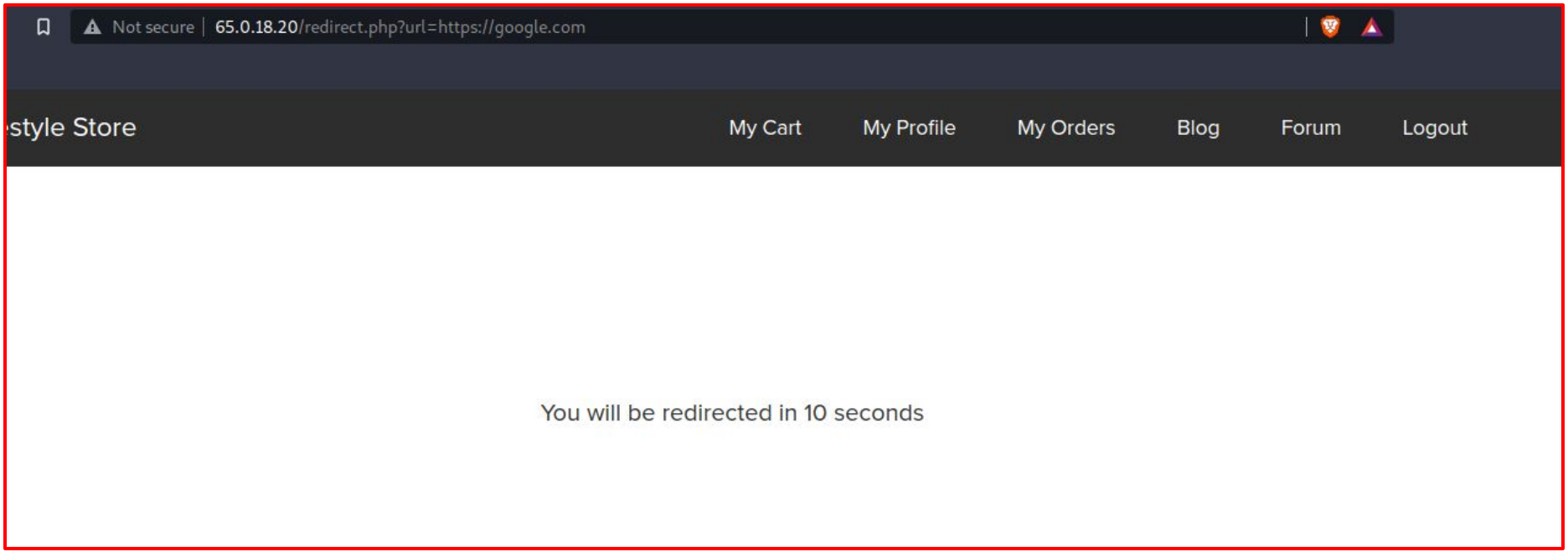
Payload : http://url.com/redirect.php?url=https://www.google.com



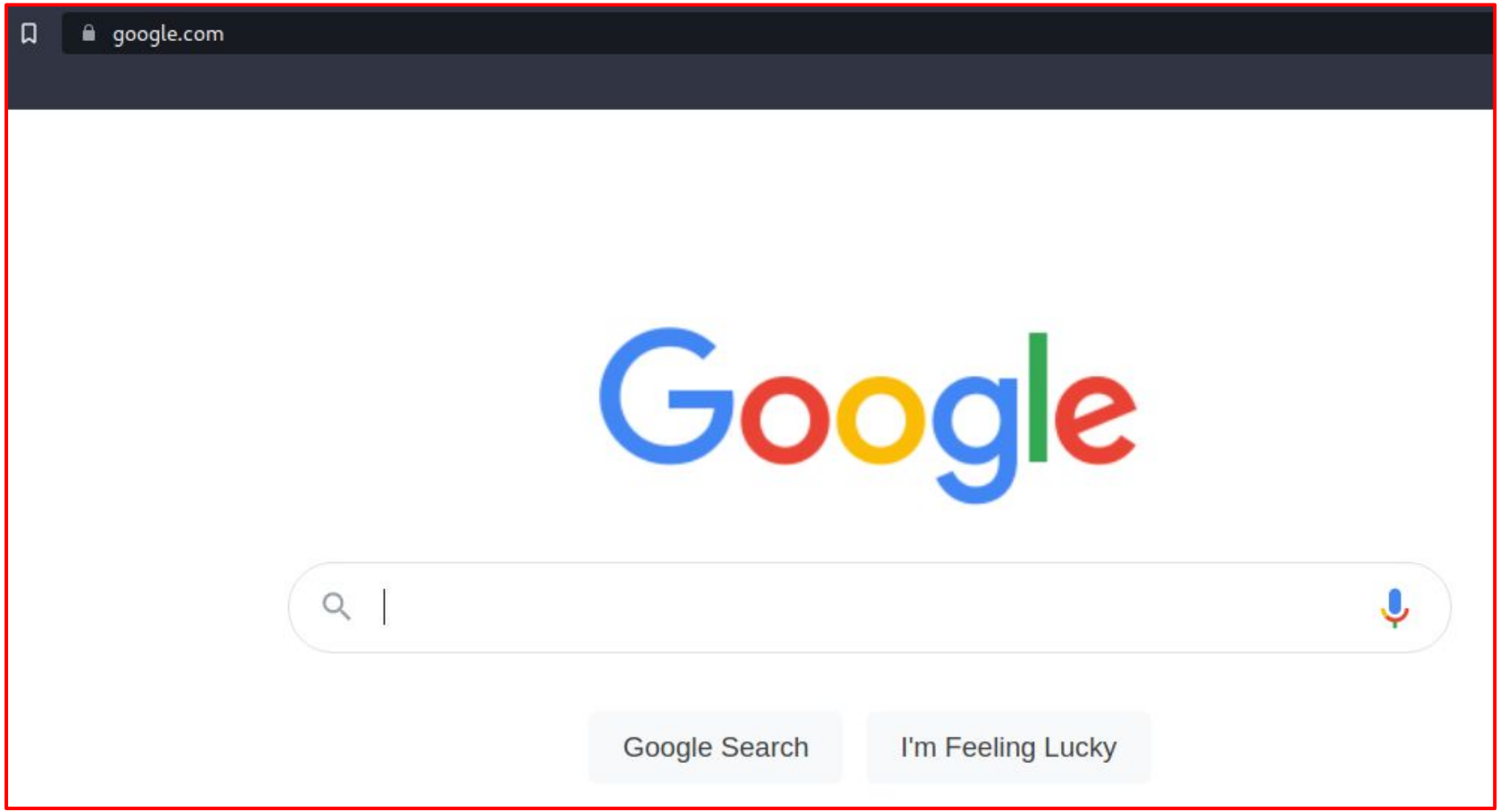


Proof of Concept (PoC)

We put http://url.com/redirect.php?url=https://www.google.com payload in the url and we redirect to new site.



Proof of Concept (PoC)



Business Impact

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

Recommendation

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

Reference

1. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client_Side_Testing/04-Testing_for_Client_Side_URL_Redirect
2. https://owasp.org/search/?searchString=open+redirection

13) Crypto Configuration Flaw

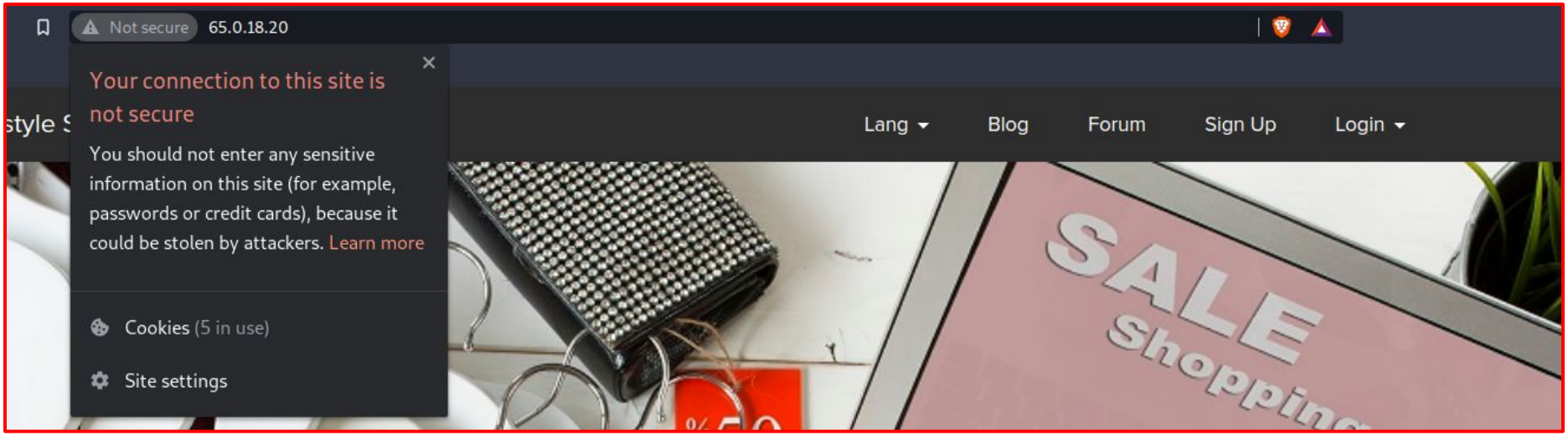
13. Crypto Configuration Flaw (severe)

Below mentioned URL is vulnerable to Crypto Configuration Flaw (Man in the Middle Attack) attack.

Affected URL :
http://url.com

Observation

All website use ‘https’ in this time but in this site ‘http’ is used means its not secure and all the request made between server and client are in plain text not encrypted.
HTTPs is encrypted and secure.



Business Impact

1. Security is almost halved in http providing easy man-in-the-middle attack and others which makes it easy for attacker to go through the data transmitted over the internet.
2. Traffic between server and client can be captured and read by attacker using Man In The Middle Attack techniques.

Recommendation

1. Use https and not http as the protocol.
2. E-Commerce like website should use HSTS for much more security.

Reference

1. https://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html
2. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security

14) Components with known vulnerability

14. Components with known vulnerability (severe)
Below mentioned URL is vulnerable to Components with known vulnerability. **Affected Components :** ● Wonder CMS ● Ovedentia CMS ● Codo Forum

Proof of Concept (POC)

WonderCMS version used in website

EXPLOIT DATABASE

Verified Has App

Filters Reset All

Show 15

Search: wondercms

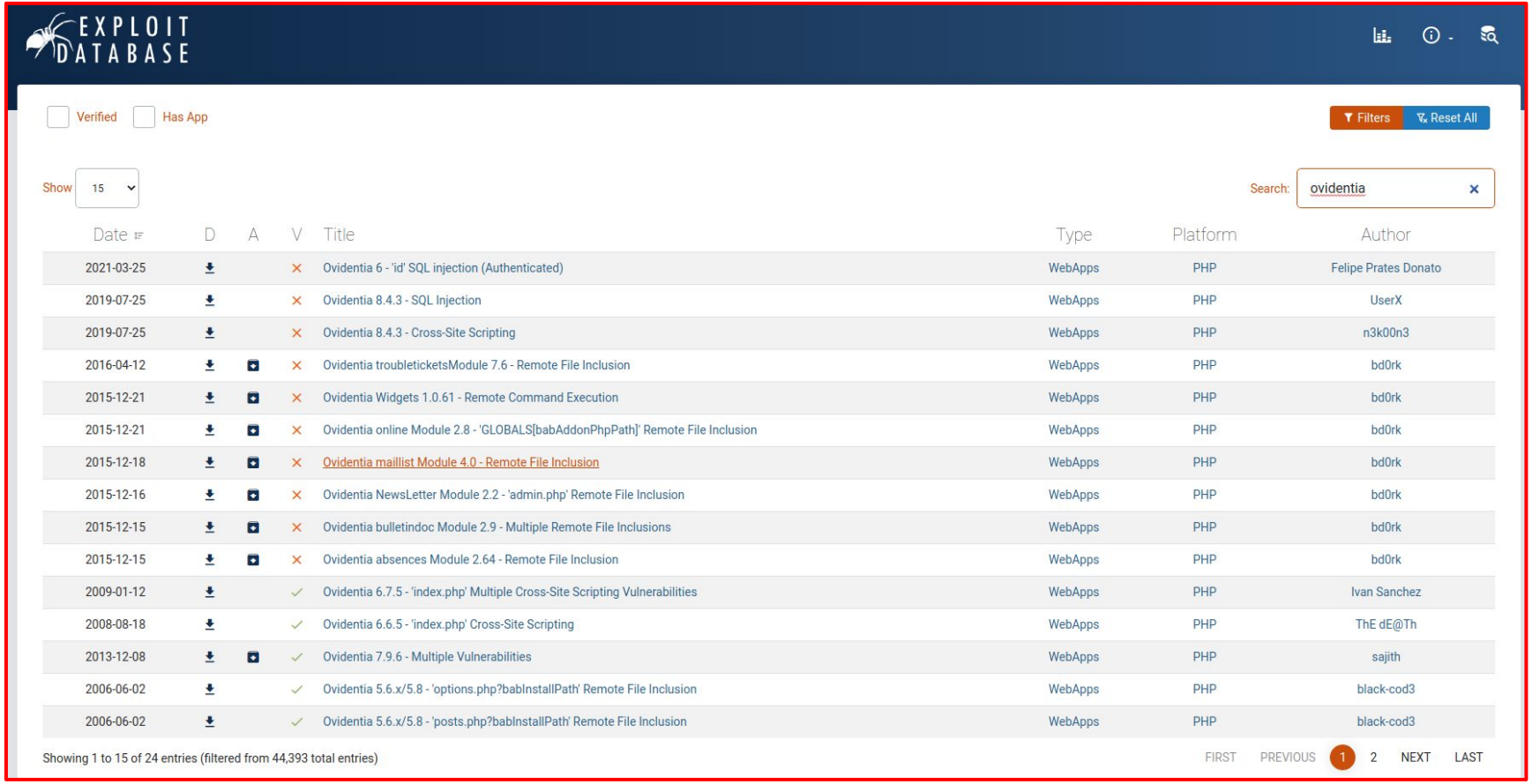
Date	D	A	V	Title	Type	Platform	Author
2020-12-02				WonderCMS 3.1.3 - 'Menu' Persistent Cross-Site Scripting	WebApps	PHP	Hemant Patidar
2020-12-02				WonderCMS 3.1.3 - Authenticated Remote Code Execution	WebApps	PHP	zetc0de
2020-12-02				WonderCMS 3.1.3 - Authenticated SSRF to Remote Remote Code Execution	WebApps	PHP	zetc0de
2020-11-27				WonderCMS 3.1.3 - 'uploadFile' Stored Cross-Site Scripting	WebApps	PHP	Sun* Cyber Security Research Team
2020-11-25				WonderCMS 3.1.3 - 'page' Persistent Cross-Site Scripting	WebApps	PHP	Mayur Parmar
2020-11-20				WonderCMS 3.1.3 - 'content' Persistent Cross-Site Scripting	WebApps	PHP	Hemant Patidar
2017-06-19				WonderCMS 2.1.0 - Cross-Site Request Forgery	WebApps	PHP	Ehsan Hosseini
2011-01-04				WonderCMS 0.3.3 - 'editText.php' Cross-Site Scripting	WebApps	PHP	High-Tech Bridge SA

Showing 1 to 8 of 8 entries (filtered from 44,393 total entries)

FIRST PREVIOUS 1 NEXT LAST

Proof of Concept (POC)

Vulnerability available for Ovidentia CMS on internet



EXPLOIT DATABASE

Verified | Has App | Filters | Reset All

Show 15 | Search: ovidentia

Date	D	A	V	Title	Type	Platform	Author
2021-03-25				Ovidentia 6 - 'id' SQL injection (Authenticated)	WebApps	PHP	Felipe Prates Donato
2019-07-25				Ovidentia 8.4.3 - SQL Injection	WebApps	PHP	UserX
2019-07-25				Ovidentia 8.4.3 - Cross-Site Scripting	WebApps	PHP	n3k00n3
2016-04-12				Ovidentia troubleticketsModule 7.6 - Remote File Inclusion	WebApps	PHP	bd0rk
2015-12-21				Ovidentia Widgets 1.0.61 - Remote Command Execution	WebApps	PHP	bd0rk
2015-12-21				Ovidentia online Module 2.8 - 'GLOBALS[babAddonPhpPath]' Remote File Inclusion	WebApps	PHP	bd0rk
2015-12-18				Ovidentia maillist Module 4.0 - Remote File Inclusion	WebApps	PHP	bd0rk
2015-12-16				Ovidentia NewsLetter Module 2.2 - 'admin.php' Remote File Inclusion	WebApps	PHP	bd0rk
2015-12-15				Ovidentia bulletindoc Module 2.9 - Multiple Remote File Inclusions	WebApps	PHP	bd0rk
2015-12-15				Ovidentia absences Module 2.64 - Remote File Inclusion	WebApps	PHP	bd0rk
2009-01-12				Ovidentia 6.7.5 - 'index.php' Multiple Cross-Site Scripting Vulnerabilities	WebApps	PHP	Ivan Sanchez
2008-08-18				Ovidentia 6.6.5 - 'index.php' Cross-Site Scripting	WebApps	PHP	ThE dE@Th
2013-12-08				Ovidentia 7.9.6 - Multiple Vulnerabilities	WebApps	PHP	sajith
2006-06-02				Ovidentia 5.6.x/5.8 - 'options.php?babInstallPath' Remote File Inclusion	WebApps	PHP	black-cod3
2006-06-02				Ovidentia 5.6.x/5.8 - 'posts.php?babInstallPath' Remote File Inclusion	WebApps	PHP	black-cod3

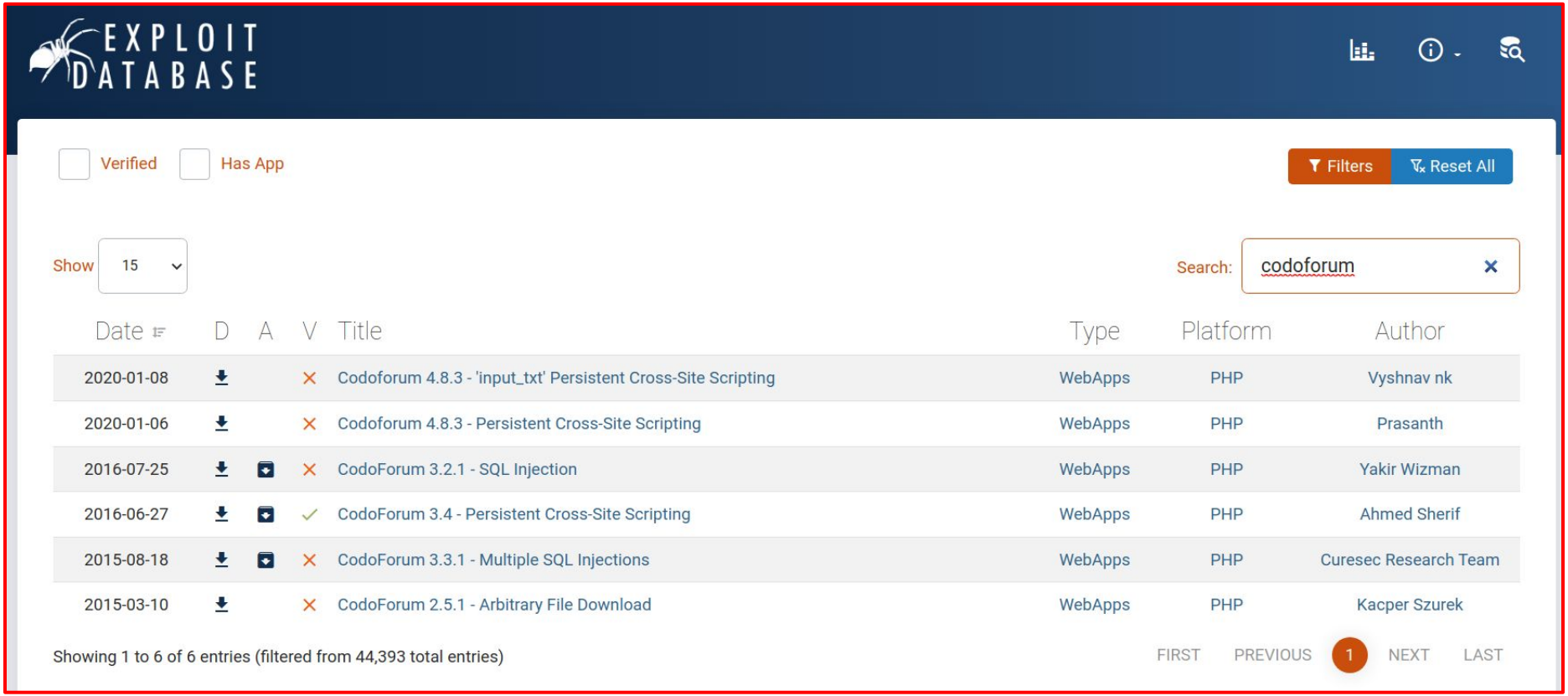
Showing 1 to 15 of 24 entries (filtered from 44,393 total entries)

FIRST PREVIOUS 1 2 NEXT LAST

Proof of Concept (POC)

© 2015 CODOLOGIC
Powered by Codoforum

Codoforum version used in website.



Business Impact – high

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

Recommendation

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

References:

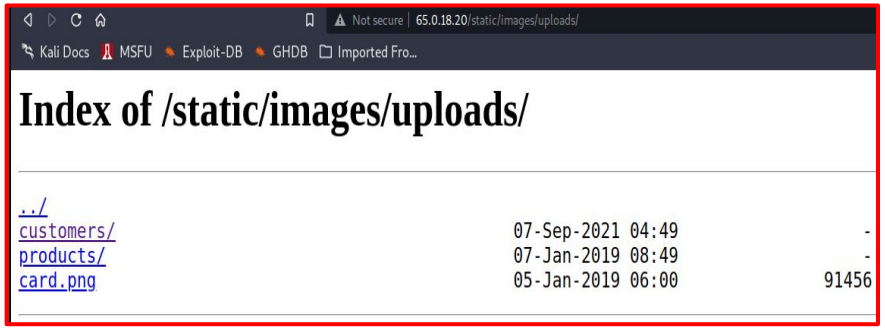
1. https://usn.ubuntu.com/4099-1/ (for ubuntu)
2. https://www.exploit-db.com/exploits/37820
3. https://securitywarrior9.blogspot.com/2018/01/vulnerability-in-wonder-cms-leading-to.html

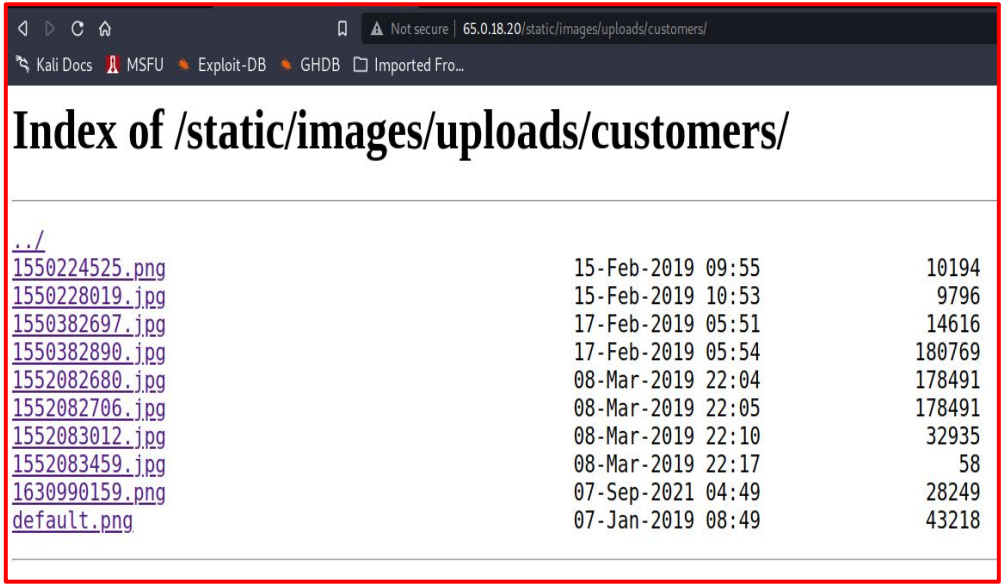
15) Directory Listing

15. Directory Listing (moderate)
Below mentioned URL is vulnerable to Confidential Directory Listing. **Affected URL :** http://url.com/static/images/uploads/customers

Observation

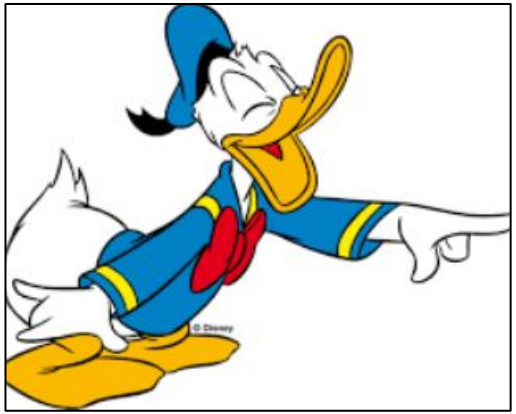
Go to this site : http://url.com/static/images/uploads/customers.
We get complete list of profile pictures of all users and pictures are using in this site.





Proof of Concept (PoC)

We get all profile pictures of all users. Some pictures are posted as a proof.



Business Impact

1. Using this vulnerability, attacker can not harm to user or the server but the attacker can steal the information of the user or website and download the backup of the website.
2. Attacker can view the image of the user and view the data of downloaded backup.

Recommendation

1. To prevent this vulnerability disable the directory listing option.
2. Put an index.html in all folders with default message.

Reference

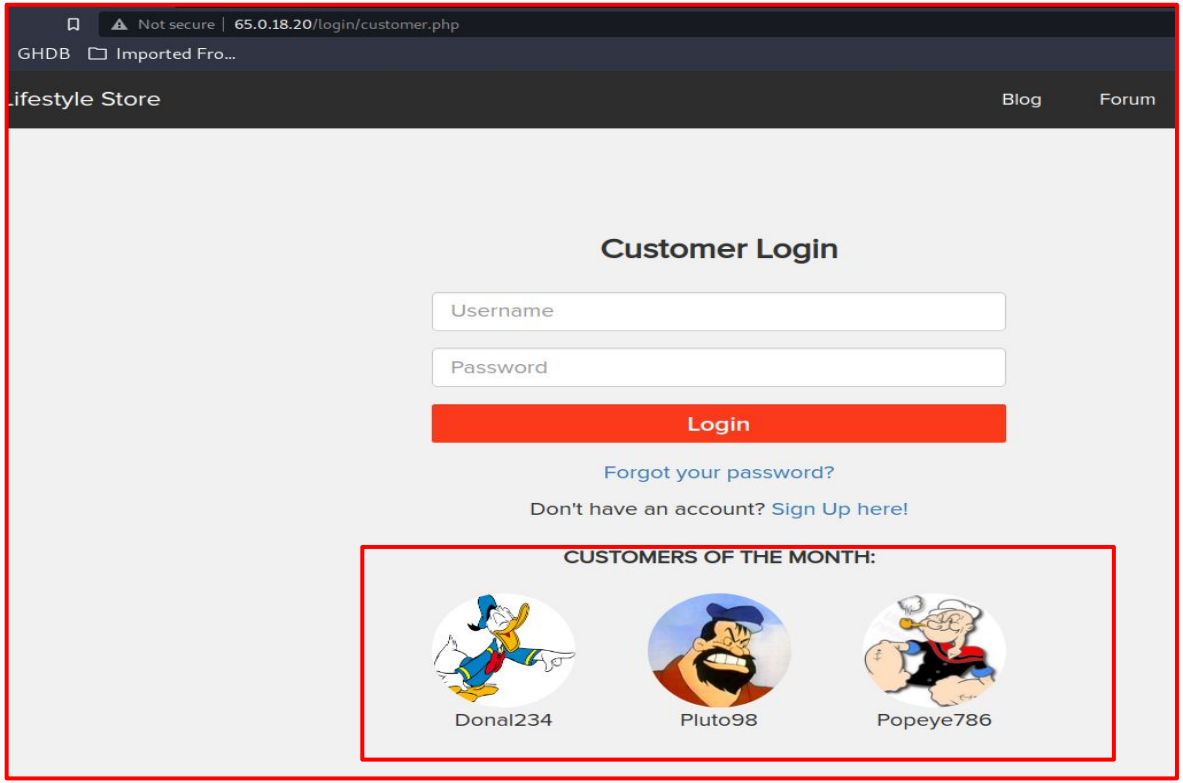
1. https://cwe.mitre.org/data/definitions/548.html
2. https://www.netsparker.com/blog/web-security/disable-directory-listing-web-servers/

16) Personally Identifiable Information Leakage (PII)

16. Personally Identifiable Information (moderate)
Below mentioned URL is vulnerable to PII Leakage. **Affected URL :** http://url.com/login/customer.php

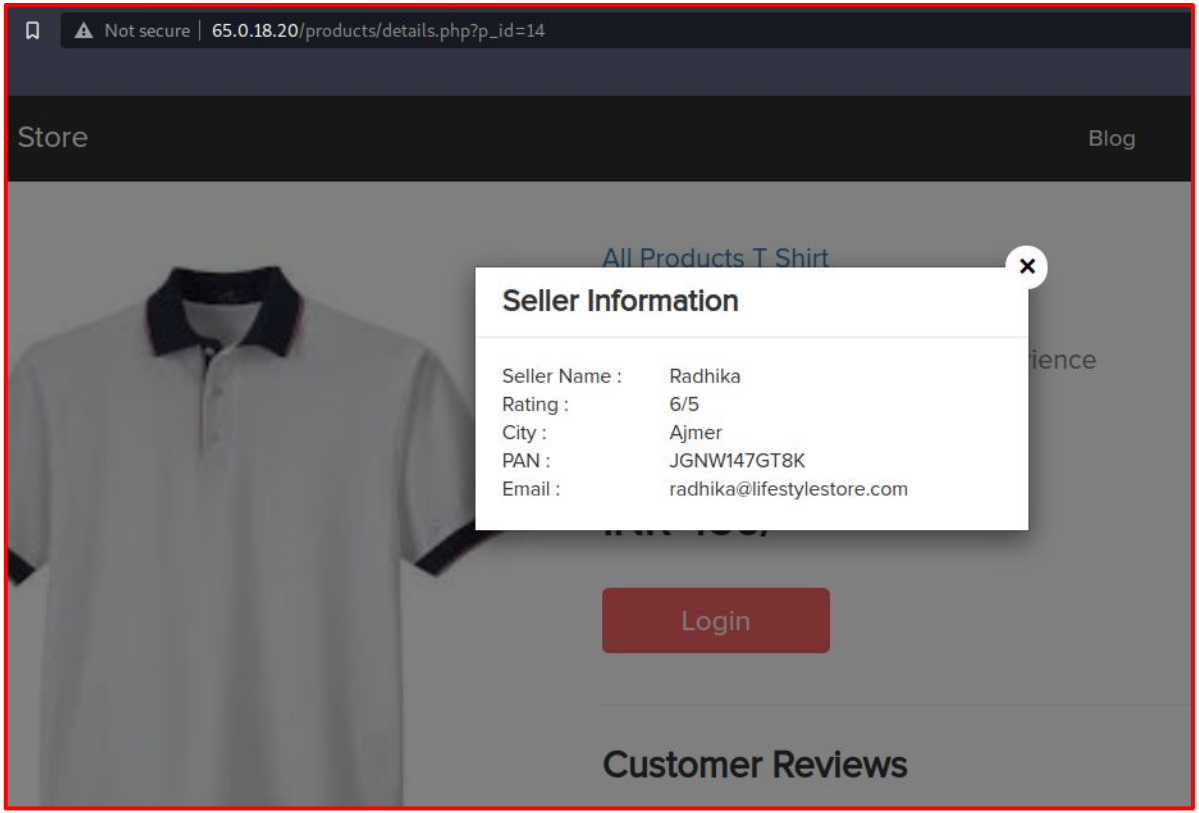
Observation

In the login page of customer we see the customer's profile picture and username are given as customer of the month.



Observation

In the products details we can see personal information about seller like PAN , Email & City.



Business Impact

1. Using this vulnerability, attacker get the username of the user then attacker can use forgot password option and change the password of user.
2. Attacker can access the account of user and get sensitive information of user.

Recommendation

1. Website can not display the name publically.
2. There are some steps for verification like reset password link on email, get OTP which is sent by site, etc to change the password of user.

Reference

1. https://cipher.com/blog/25-tips-for-protecting-pii-and-sensitive-data/
2. https://digitalguardian.com/blog/how-secure-personally-identifiable-information-against-loss-or-compromise

17) Outdated version of using components

17. Outdated version of using components (moderate)

Below mentioned website components are vulnerable to Outdated version of using components.

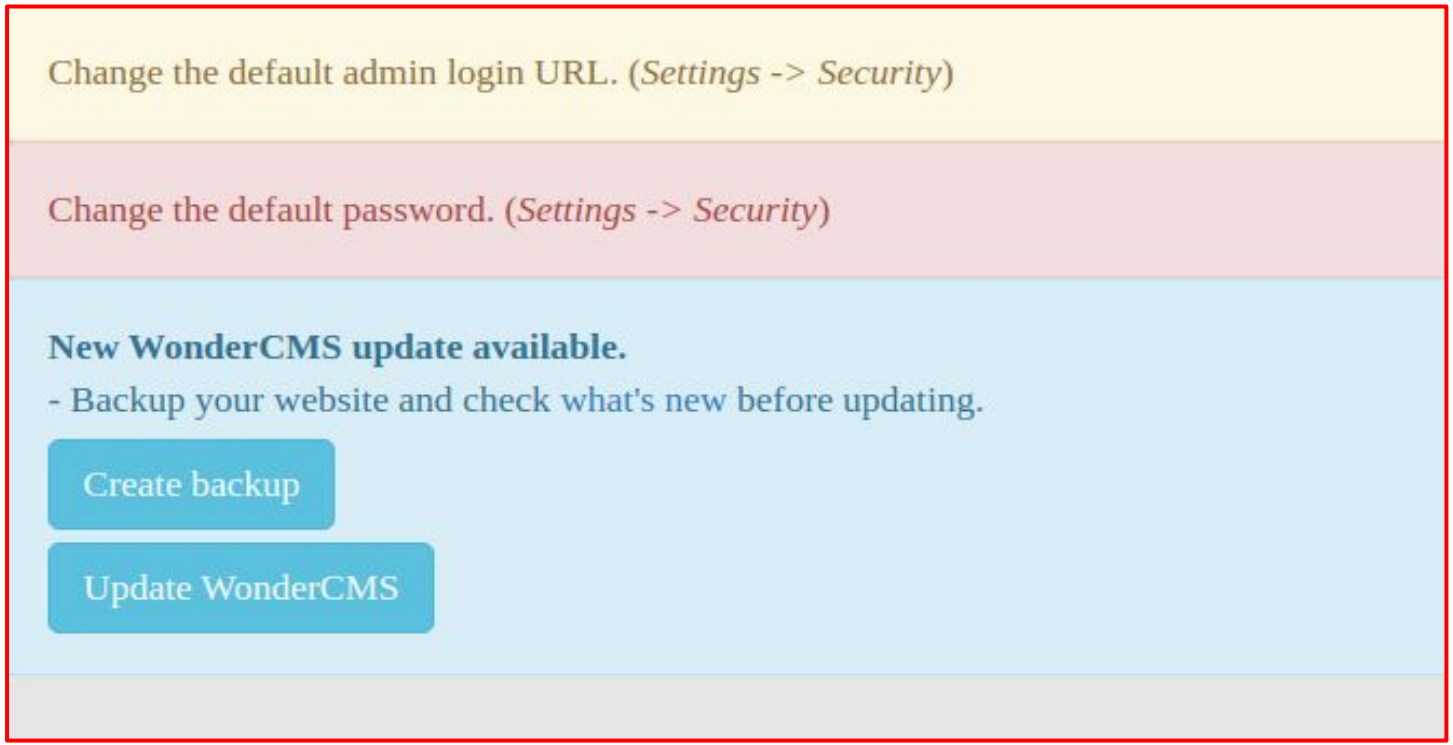
Affected Components:
PHP
Wonder CMS

Observation

Using PHP version in this site is outdated not to be use latest version of PHP.
Latest version of PHP is 8.0.10 but in this lifestyle site using 5.6.39 version.

Observation

This site using WonderCMS to post blog and post the content of the website. Using WonderCMS is very outdated version at this time we need to update WonderCMS into new version.



Business Impact

1. Attacker can easily attack on outdated versions of software because outdated software has some vulnerability so, the attacker can search any vulnerability is available in this software. Attacker exploits that vulnerability and attack on site.

Recommendation

1. To prevent this vulnerability we should use latest and updated software to build website.

Reference

1. https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities
2. https://www.tutorialspoint.com/security_testing/components_with_vulnerabilities.htm
3. https://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/version_id-298515/PHP-PHP-5.6.39.html

18) Server side & Client side misconfiguration flaw

18. Server side & Client side misconfiguration flaw (Low)

Below mentioned website components are vulnerable to missing server-side validation.

Affected URL:
http://url.com/forum/index.php?u=/user/register

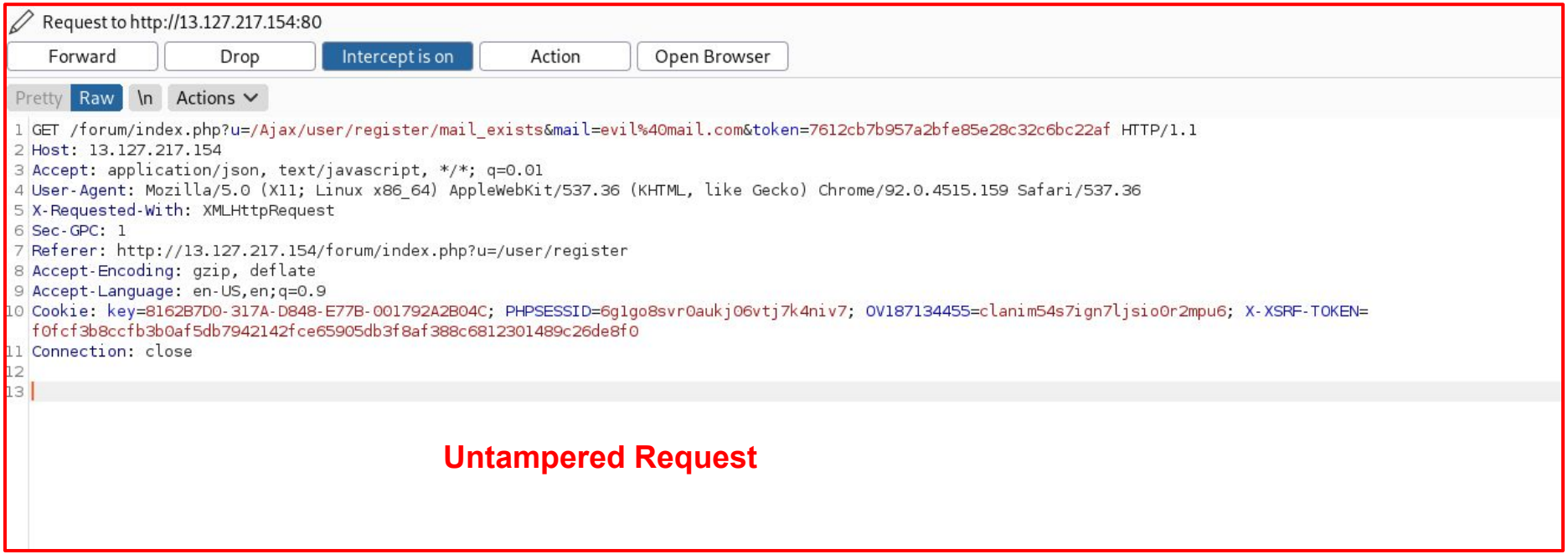
Affected Parameter :
Email address

Affected URL :
http://url.com/profile/16/edit

Affected Parameter :
Phone

Observation

When you sign up in forum panel you must enter proper email address. We intercept the request and tamper with the email address in burpsuite. In repeater we check the output with tampered email address.



Observation

Request to http://13.127.217.154:80

Forward | Drop | Intercept is on | Action | Open Browser

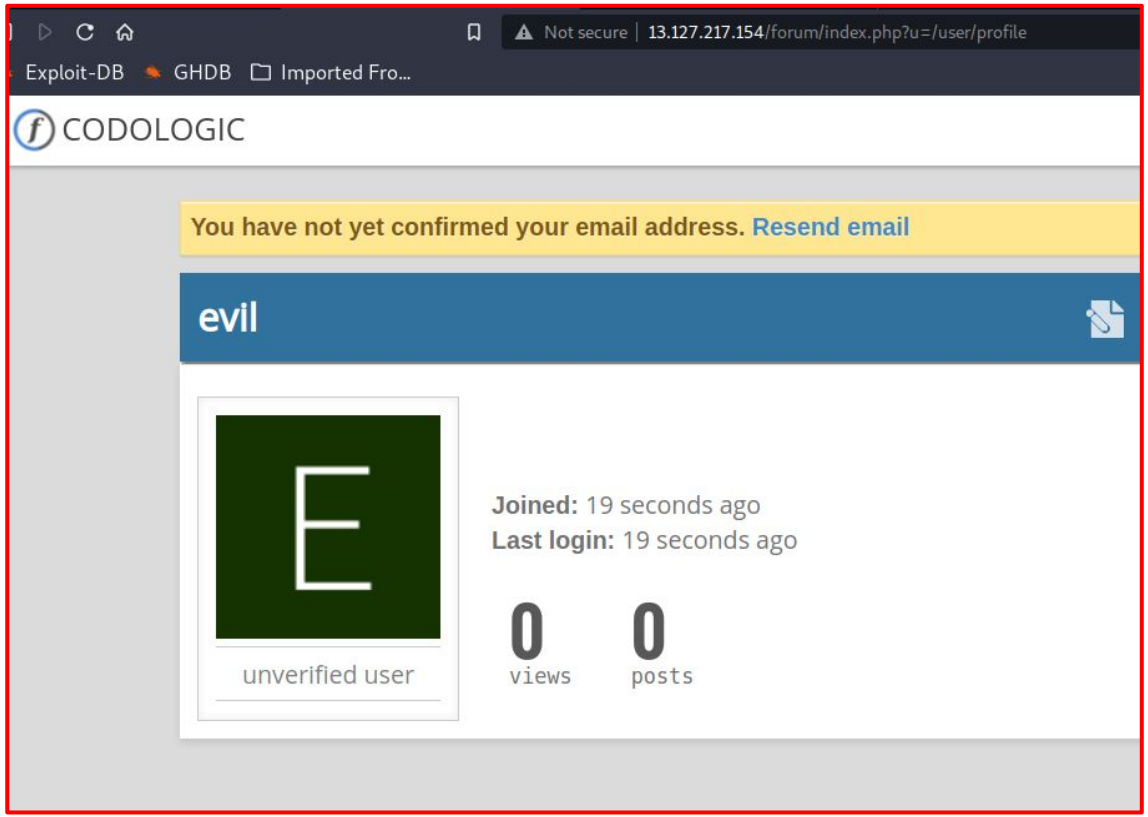
Pretty | Raw | \n | Actions

```
1  GET /forum/index.php?u=/Ajax/user/register/mail_exists&mail=evil&token=7612cb7b957a2bfe85e28c32c6bc22af HTTP/1.1
2  Host: 13.127.217.154
3  Accept: application/json, text/javascript, */*; q=0.01
4  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36
5  X-Requested-With: XMLHttpRequest
6  Sec-GPC: 1
7  Referer: http://13.127.217.154/forum/index.php?u=/user/register
8  Accept-Encoding: gzip, deflate
9  Accept-Language: en-US,en;q=0.9
10 Cookie: key=8162B7D0-317A-D848-E77B-001792A2B04C; PHPSESSID=6g1go8svr0aukj06vtj7k4niv7; OV187134455=clanim54s7ign7ljsio0r2mpu6; X-XSRF-TOKEN=
   f0fcf3b8ccfb3b0af5db7942142fce65905db3f8af388c6812301489c26de8f0
11 Connection: close
12
13
```

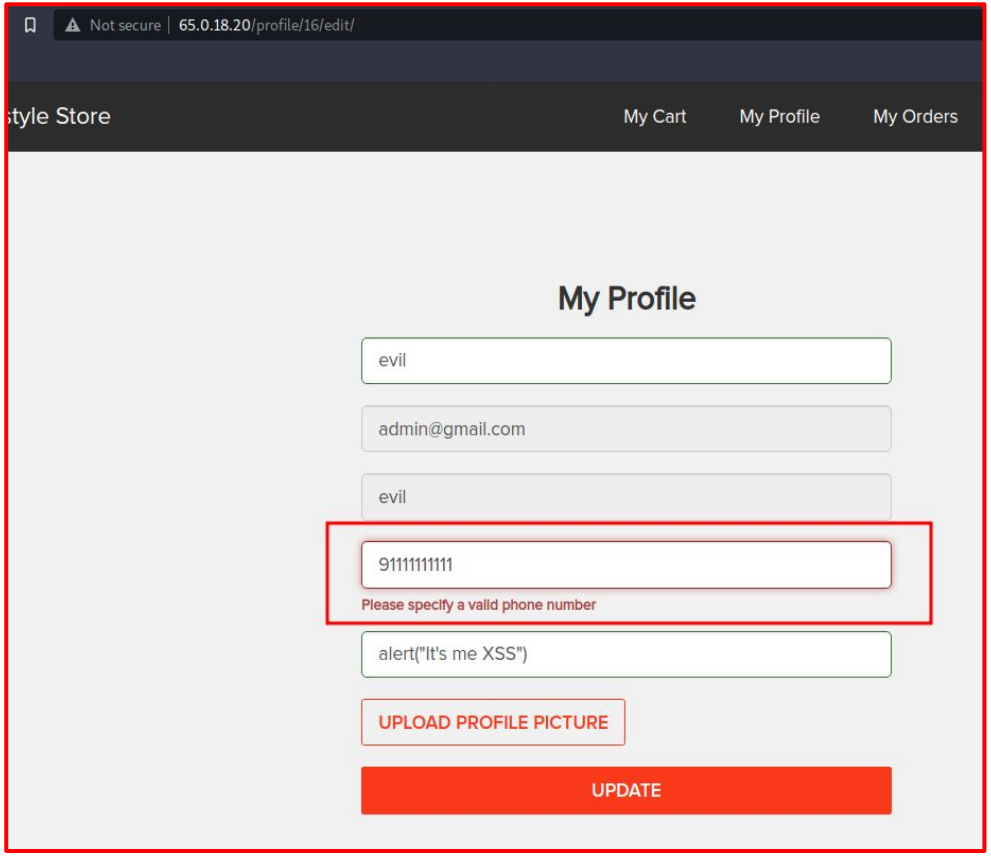
Tampered Request (Email Changed)

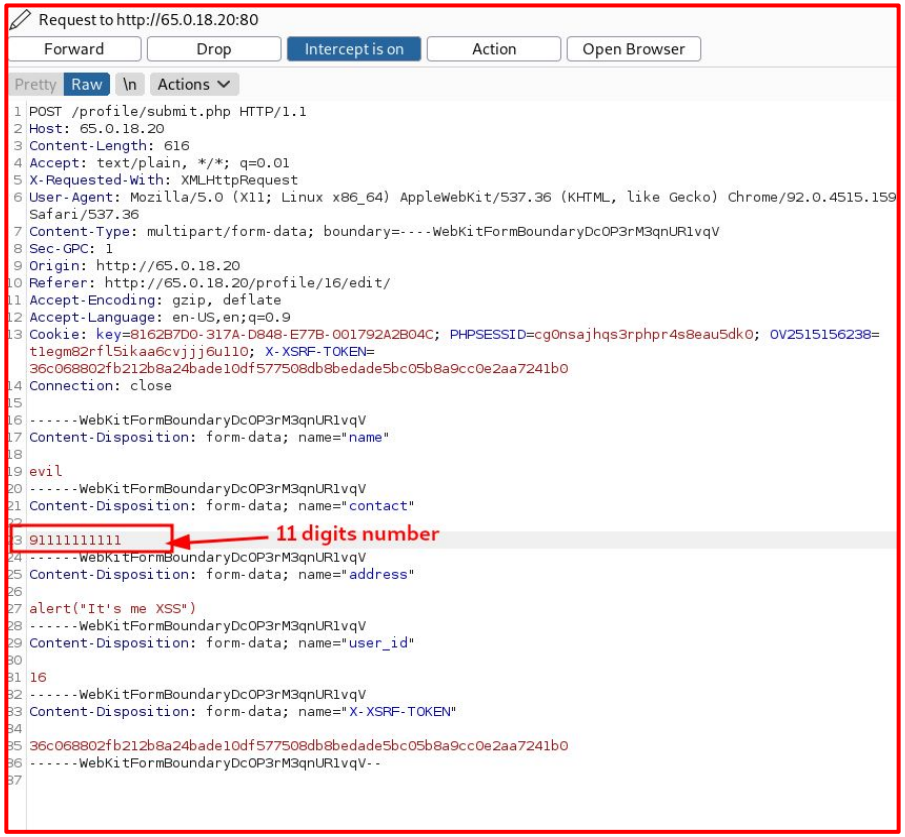
Proof of Concept (PoC)

We drop the original request and refresh the page we created account successfully in forum.



Observation





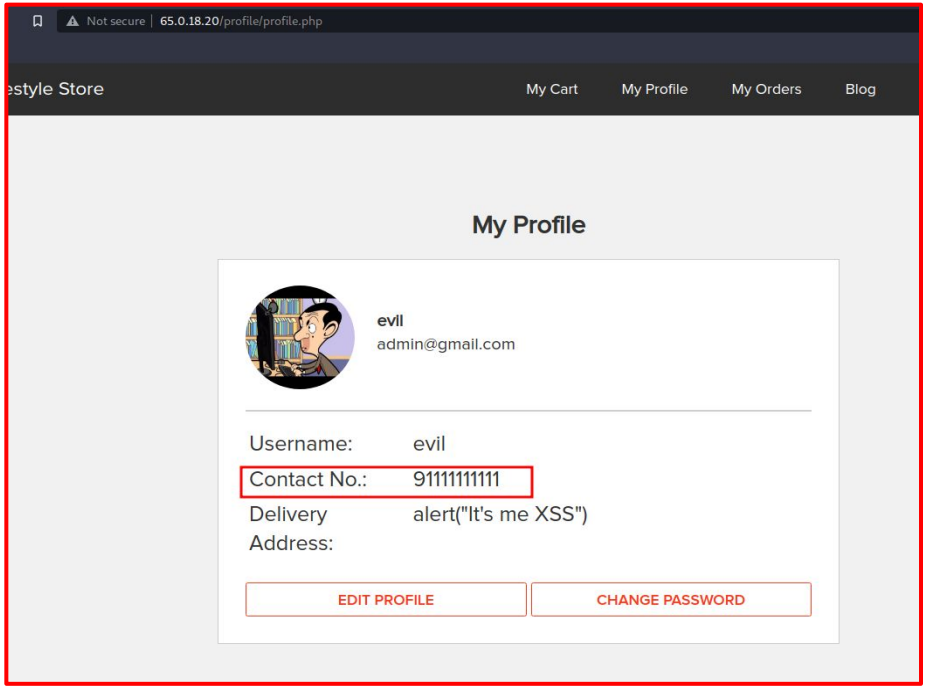
Proof of Concept (POC)

Response from http://65.0.18.20:80/profile/submit.php

Forward | Drop | Intercept is on | Action | Open Browser

Pretty | Raw | Render | \n | Actions

```
1  HTTP/1.1 200 OK
2  Server: nginx/1.14.0 (Ubuntu)
3  Date: Tue, 07 Sep 2021 14:06:28 GMT
4  Content-Type: text/html; charset=utf-8
5  Connection: close
6  Expires: Thu, 19 Nov 1981 08:52:00 GMT
7  Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
8  Pragma: no-cache
9  X-FRAME-OPTIONS: DENY
10 Set-Cookie: X-XSRF-TOKEN=58928c1bd1858e44f64d58c7d1098b1319a02651b2852f18bb61af825c4c502f;
11 Content-Length: 64
12
13 {"success":true,"successMessage":"Profile updated succesfully."}
```



Business Impact

1. The data provided by the user ,if data is incorrect that's not a very big issue but still must be checked for proper validation.

Recommendation

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

Reference

1. https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A6-Security_Misconfiguration
2. https://www.owasp.org/index.php/Unvalidated_Input

19) Descriptive error message

19. Descriptive error message (Low)

Below mentioned website components are vulnerable to Descriptive error message.

Affected URL:
http://url.com/?includelang=lang/en.php

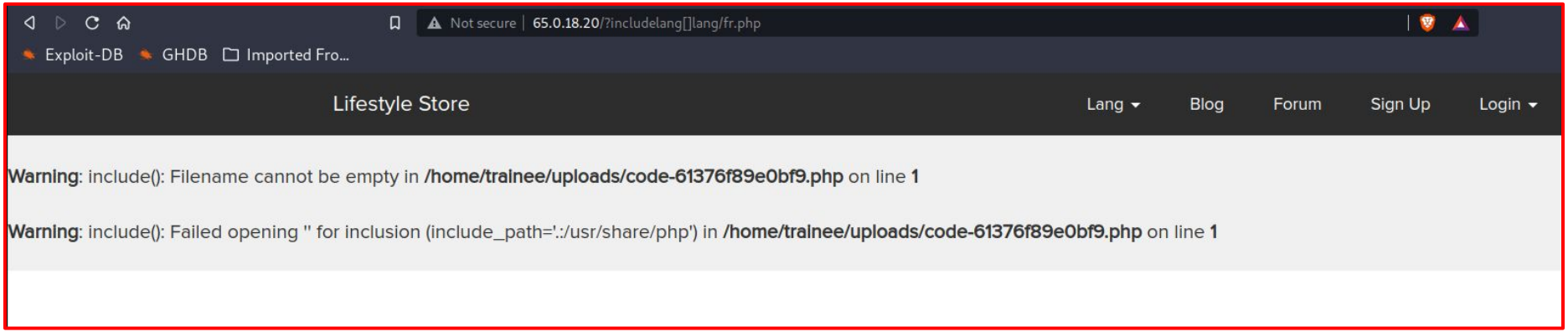
Payload :
Includelang[ ]=lang/en.php

Affected URL :
http://url.com/products.php?cat=1’

Payload :
cat=1’

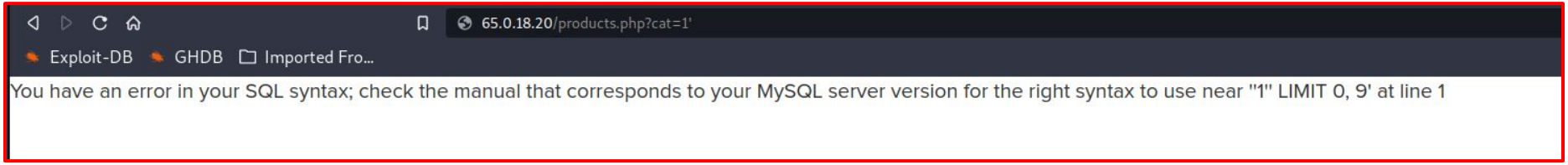
Observation

We click on language and include some special character in the url of language. It display some error that provide some information about the server.
Payload URL : url.com/?includelang[ ]=lang/fr.php



Observation

When we search some product in search bar and include some special character in GET based url. It display some error.
Payload URL : url.com/products.php?cat=1’



Business Impact

1. This type of vulnerability is not direct impact on user or server but this is use to attacker mapping the architecture of a website and plan further attack on server.

Recommendation

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

Reference

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

20) Default files/pages

20. Default files/pages (Low)

Below mentioned website components are vulnerable to Default files/pages.

Default files :

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

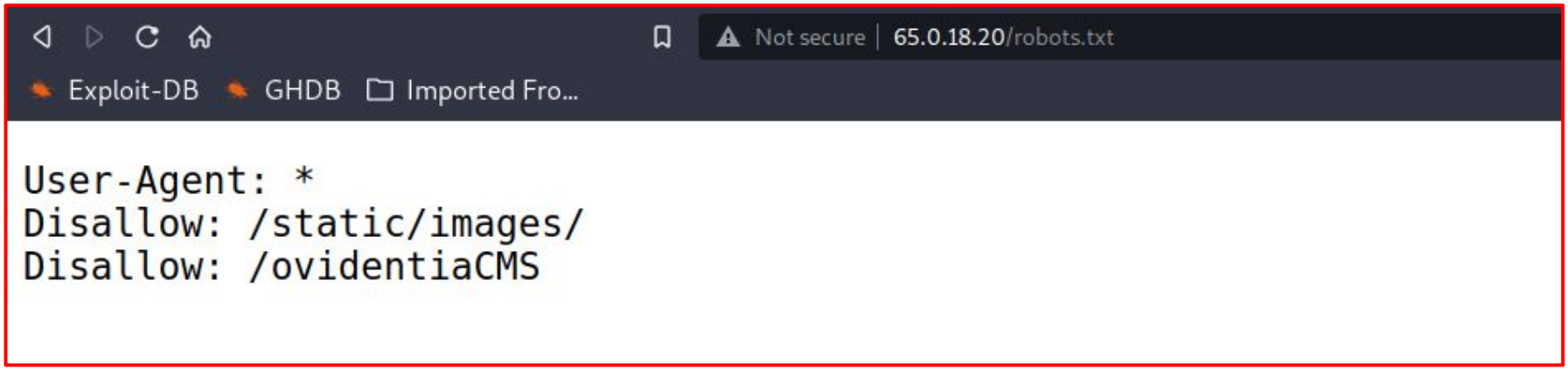
Observation

When we used gobuster with common.txt wordlist we got many default files and pages .

```
┌──(root💀kali)-[~]
└─# gobuster dir -u http://13.127.217.154/  -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://13.127.217.154/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2021/09/08 12:33:04 Starting gobuster in directory enumeration mode
===============================================================
/.htpasswd            (Status: 403) [Size: 178]
/.hta                 (Status: 403) [Size: 178]
/.htaccess            (Status: 403) [Size: 178]
/cart                 (Status: 301) [Size: 194] [--> http://13.127.217.154/cart/]
/common               (Status: 301) [Size: 194] [--> http://13.127.217.154/common/]
/config               (Status: 301) [Size: 194] [--> http://13.127.217.154/config/]
Progress: 1147 / 4615 (24.85%)                                          Progress: 1513 / 4615 (32.78%)
          /forum              (Status: 301) [Size: 194] [--> http://13.127.217.154/forum/]
/index.php            (Status: 200) [Size: 751]
/lang                 (Status: 301) [Size: 194] [--> http://13.127.217.154/lang/]
/login                (Status: 301) [Size: 194] [--> http://13.127.217.154/login/]
/orders               (Status: 301) [Size: 194] [--> http://13.127.217.154/orders/]
/phpinfo.php          (Status: 200) [Size: 90344]
/products             (Status: 301) [Size: 194] [--> http://13.127.217.154/products/]
/profile              (Status: 301) [Size: 194] [--> http://13.127.217.154/profile/]
/robots.txt           (Status: 200) [Size: 65]
/search               (Status: 301) [Size: 194] [--> http://13.127.217.154/search/]
/server-status        (Status: 301) [Size: 194] [--> http://13.127.217.154/server-status/]
/signup               (Status: 301) [Size: 194] [--> http://13.127.217.154/signup/]
/static               (Status: 301) [Size: 194] [--> http://13.127.217.154/static/]
/vendor               (Status: 301) [Size: 194] [--> http://13.127.217.154/vendor/]

===============================================================
2021/09/08 12:33:11 Finished
===============================================================
```

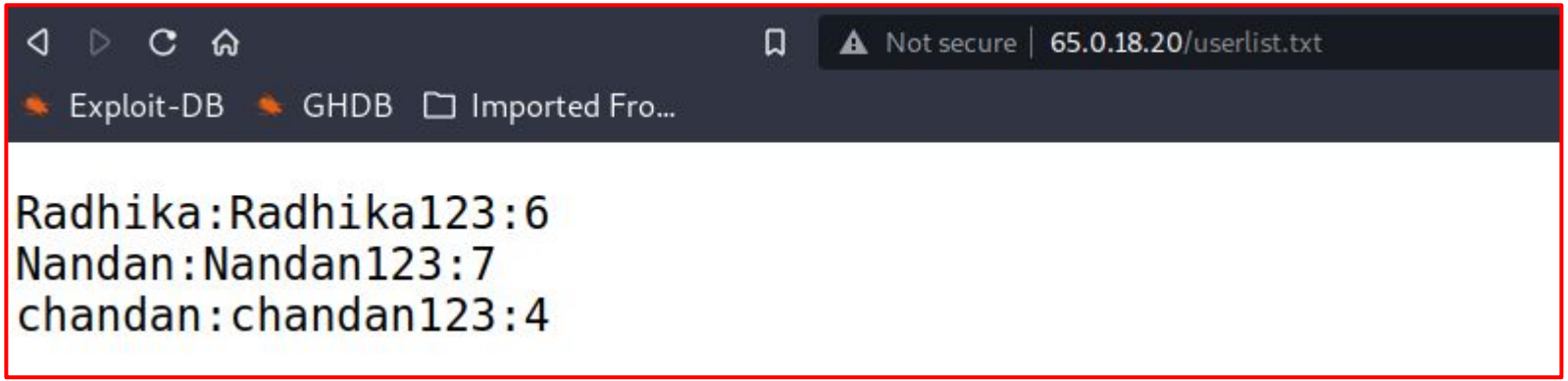
Proof of Concept (PoC)

We entered robots.txt at the end of the index page URL, We got this page.



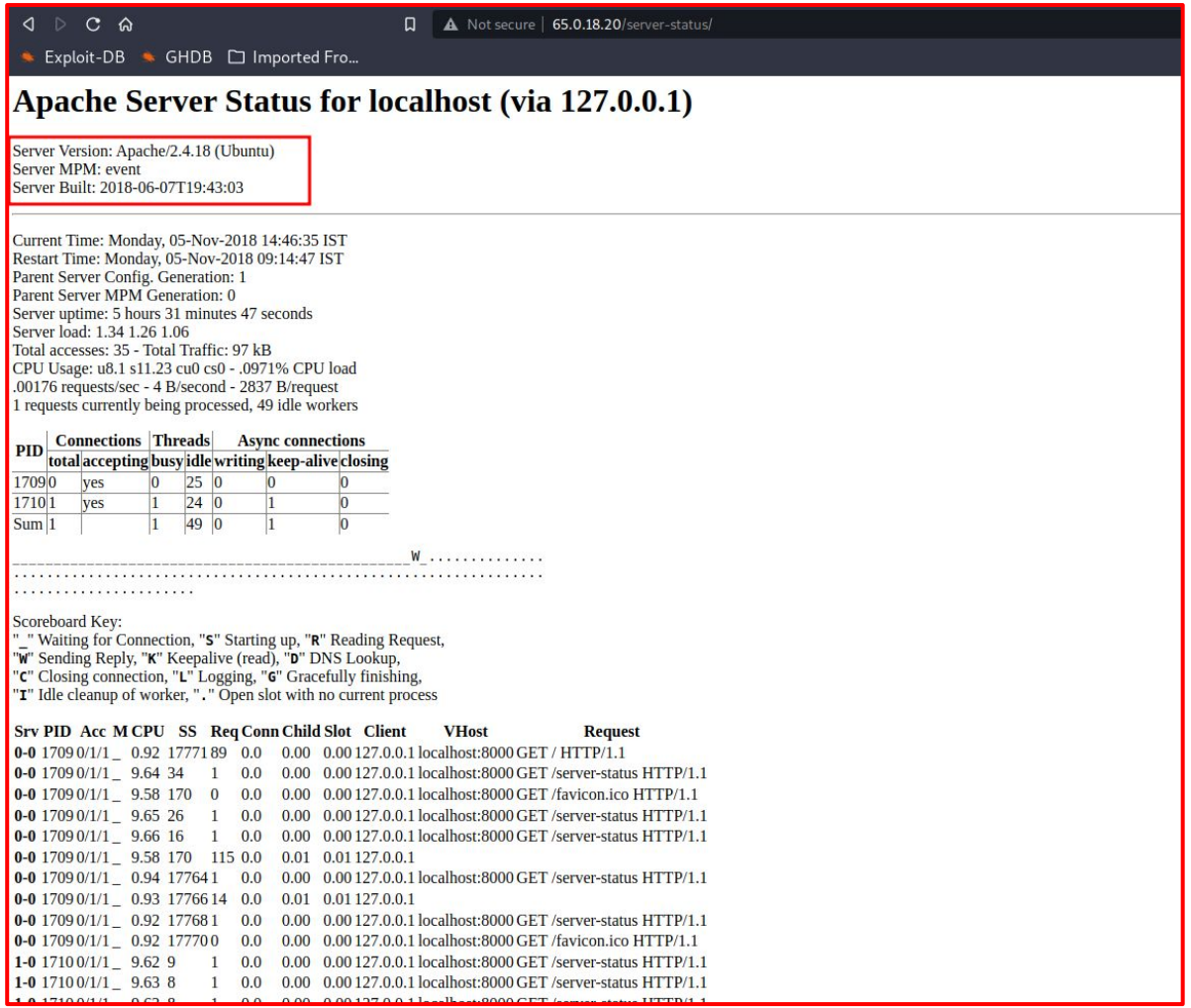
Proof of Concept (PoC)

We entered userlist.txt at the end of index page URL and we got this page.



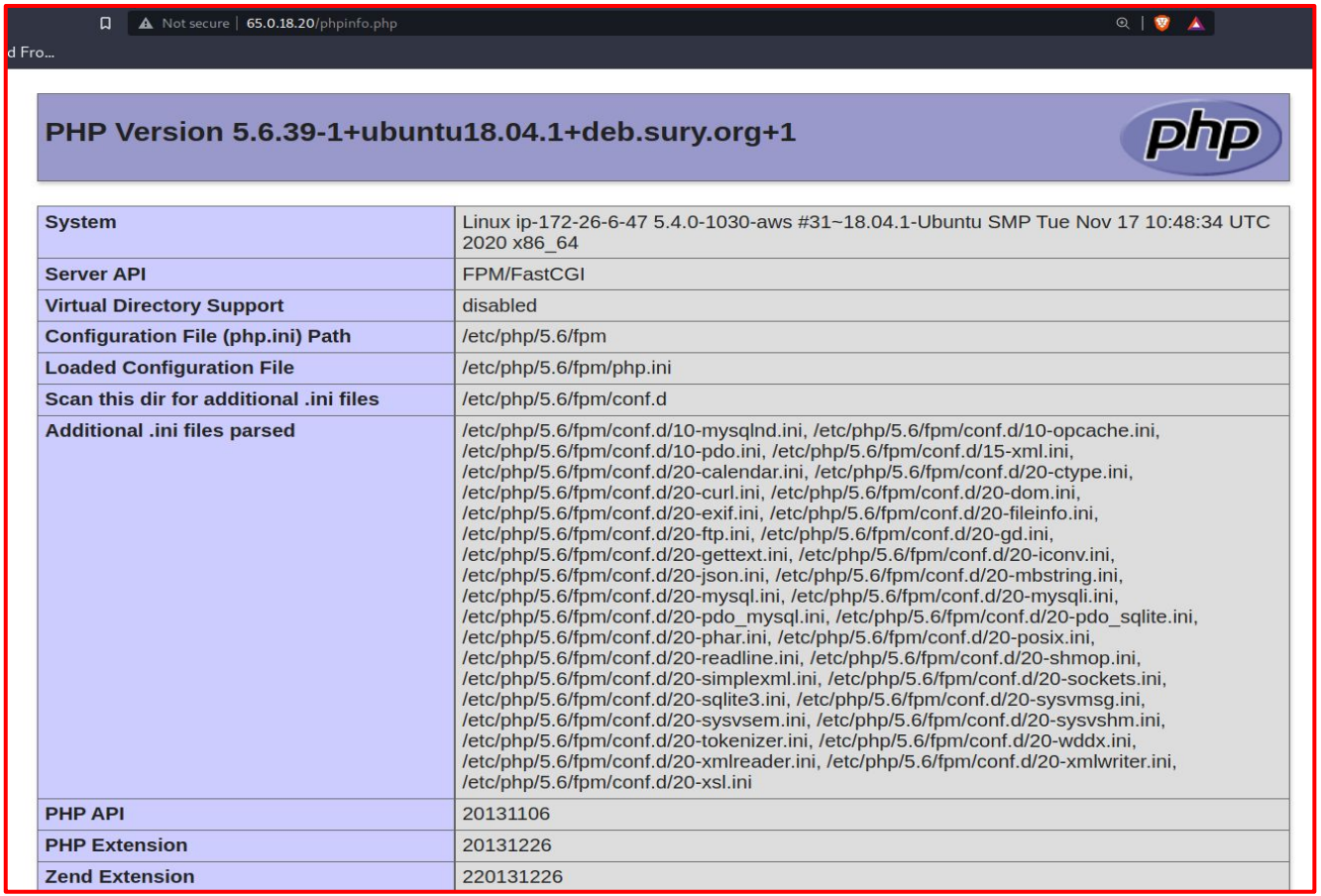
Proof of Concept(PoC)

We entered server-status at the end of index page URL and we got this page which is gives you information about the server that can be used by this website.



Proof of Concept (PoC)

We entered phpinfo.php at the end of index page URL and we get the information about the php which is used by this website.



System	Linux ip-172-26-6-47 5.4.0-1030-aws #31~18.04.1-Ubuntu SMP Tue Nov 17 10:48:34 UTC 2020 x86_64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/5.6/fpm
Loaded Configuration File	/etc/php/5.6/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/5.6/fpm/conf.d
Additional .ini files parsed	/etc/php/5.6/fpm/conf.d/10-mysqlnd.ini, /etc/php/5.6/fpm/conf.d/10-opcache.ini, /etc/php/5.6/fpm/conf.d/10-pdo.ini, /etc/php/5.6/fpm/conf.d/15-xml.ini, /etc/php/5.6/fpm/conf.d/20-calendar.ini, /etc/php/5.6/fpm/conf.d/20-ctype.ini, /etc/php/5.6/fpm/conf.d/20-curl.ini, /etc/php/5.6/fpm/conf.d/20-dom.ini, /etc/php/5.6/fpm/conf.d/20-exif.ini, /etc/php/5.6/fpm/conf.d/20-fileinfo.ini, /etc/php/5.6/fpm/conf.d/20-ftp.ini, /etc/php/5.6/fpm/conf.d/20-gd.ini, /etc/php/5.6/fpm/conf.d/20-gettext.ini, /etc/php/5.6/fpm/conf.d/20-iconv.ini, /etc/php/5.6/fpm/conf.d/20-json.ini, /etc/php/5.6/fpm/conf.d/20-mbstring.ini, /etc/php/5.6/fpm/conf.d/20-mysql.ini, /etc/php/5.6/fpm/conf.d/20-mysqli.ini, /etc/php/5.6/fpm/conf.d/20-pdo_mysql.ini, /etc/php/5.6/fpm/conf.d/20-pdo_sqlite.ini, /etc/php/5.6/fpm/conf.d/20-phar.ini, /etc/php/5.6/fpm/conf.d/20-posix.ini, /etc/php/5.6/fpm/conf.d/20-readline.ini, /etc/php/5.6/fpm/conf.d/20-shmop.ini, /etc/php/5.6/fpm/conf.d/20-simplexml.ini, /etc/php/5.6/fpm/conf.d/20-sockets.ini, /etc/php/5.6/fpm/conf.d/20-sqlite3.ini, /etc/php/5.6/fpm/conf.d/20-sysvmsg.ini, /etc/php/5.6/fpm/conf.d/20-sysvsem.ini, /etc/php/5.6/fpm/conf.d/20-sysvshm.ini, /etc/php/5.6/fpm/conf.d/20-tokenizer.ini, /etc/php/5.6/fpm/conf.d/20-wddx.ini, /etc/php/5.6/fpm/conf.d/20-xmlreader.ini, /etc/php/5.6/fpm/conf.d/20-xmlwriter.ini, /etc/php/5.6/fpm/conf.d/20-xsl.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226

Business Impact

1. This type of vulnerability is not direct impact on user or server but this is use to attacker mapping the architecture of a website and plan further attack on server.

Recommendation

1. Disable all default files and pages.

Reference

1. https://vuldb.com/?id.88482
2. https://www.beyondsecurity.com/scan_pentest_network_vulnerabilities_apache_http_server_httponly_cookie_information_disclosure

THANK YOU

For any further clarifications/patch assistance, please

Email : hrushikeshpawar9@gmail.com