Lifestyle Store Web Application

Detailed Developer Report

Security Status – Extremely Vulnerable

- Hacker can steal all the records in the Lifestyle Store server databases.
- Hacker can take control of the server and modify the files and folders in the server and access the databases.
- Hacker can inject client side code into the website and trick the users into changing how the site looks to steal the information or spoil the name of the website.
- Hacker can extract the mobile numbers, e-mail ids, PAN card numbers and addresses of the users.
- Hacker can access the admin dashboard and console, thus changing the price range and items in the website.
- Hacker trick the users into giving their crucial information by redirecting them to a malicious website.

Vulnerability Statistics

Vulnerabilities

No.	Severity	Vulnerability	Count
1	Critical	SQL Injection	2
2	Critical	Access to admin panel	2
3	Critical	Admin Account takeover via OTP bypass	1
4	Critical	Unauthorized access to customer details via IDOR and Rate Limiting Flaws	2
5	Critical	Arbitrary File Upload	1
6	Critical	Components with known Vulnerabilities	4
7	Critical	Cross Site Request Forgery	1
8	Severe	Reflected Cross Site Scripting	1
9	Severe	Open Redirection (Phishing)	1
10	Severe	PII Leakage and Forced Browsing	2
11	Moderate	Directory Listing of Configuration Files	2
12	Moderate	Default Files and Pages	1
13	Low	Information Disclosure	1

1. SQL Injection

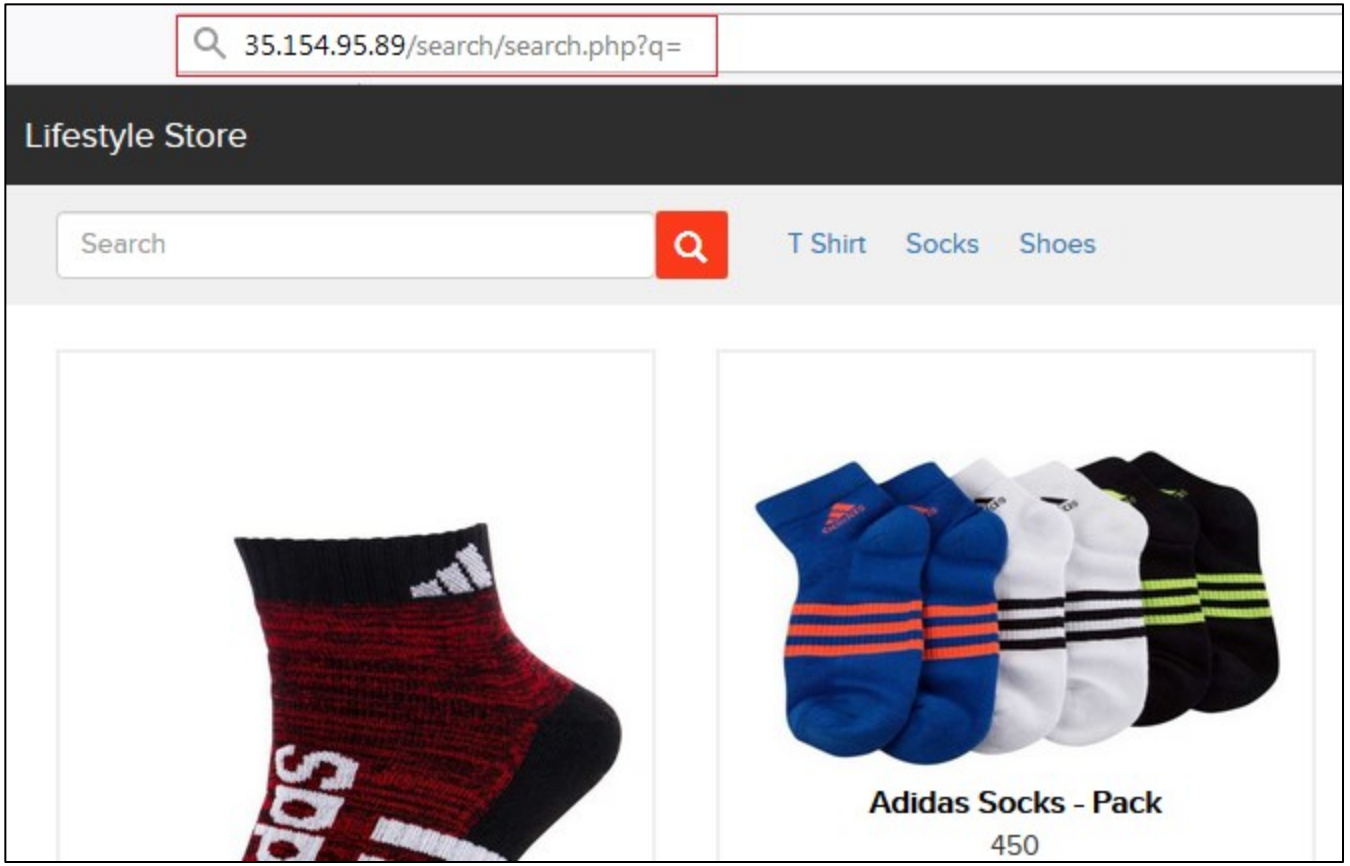
SQL Injection (Critical)	Below mentioned URL in the **Lifestyle Store section** is vulnerable to SQL Injection attack **Affected URL:** ➢ http://url/search/search.php?q=HERE **Affected Parameter:** ➢ q (GET Parameter) **Payload:** ➢ q=‘

1. SQL Injection

SQL Injection (Critical)	Here is another similar SQLi in the **Life Style Store section** **Affected URL:** ➢ http://url/products.php?cat=1 **Affected Parameter:** ➢ cat (GET Parameter) **Payload:** ➢ cat=1‘

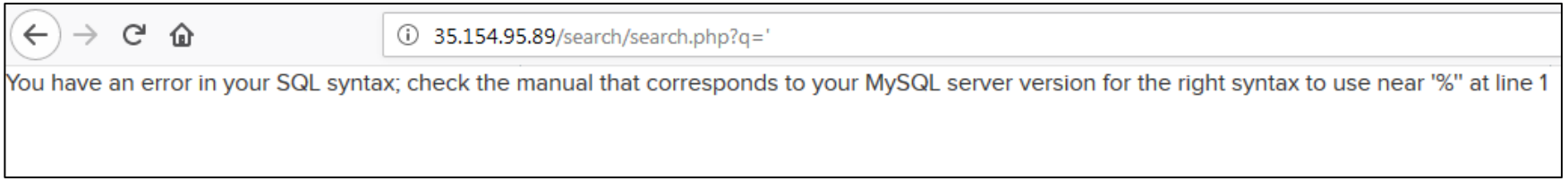
Observation

- Click on the ‘Shop Now’ button and see the detailed list of products, observe the search bar given. Do an empty search and check the GET based parameter ‘q’ in the URL.



Observation

- Now we apply a single quote at the end of the parameter ‘q’ as: **http://url/search/search.php?q=‘** and we are presented with an error as follows.

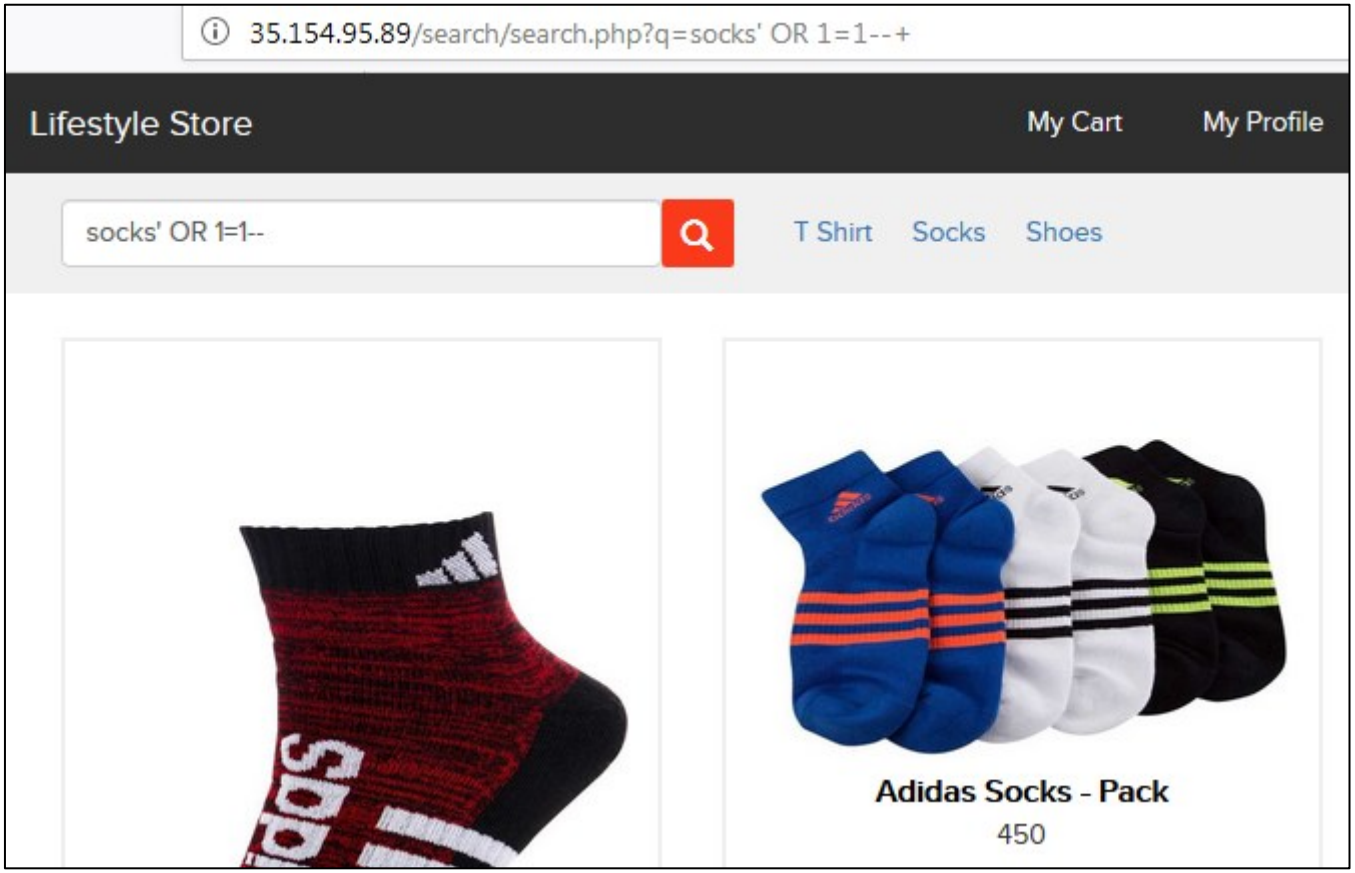


- Now we apply: ‘--+ to the URL as: **http://url/search/search.php?q=‘--+**

- The changes made to the GET parameter are reflected back in the search bar as the error is resolved now. Thus the parameter ‘q’ is injectable.

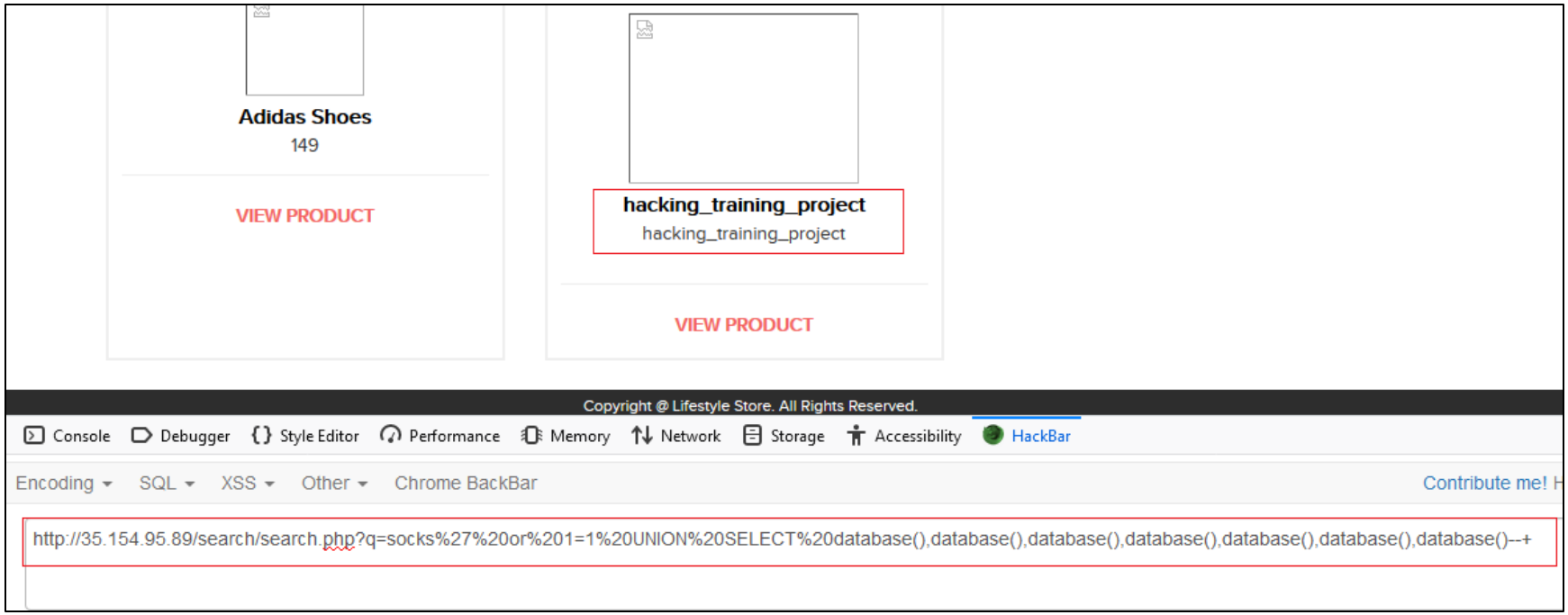
Proof Of Concept (PoC)

- To test for **PoC** we write the following statement: **http://url/search/search.php?q=socks‘ OR 1=1--+** which gives us the details about all the products (socks, t-shirts and shoes)



Proof Of Concept (PoC)

- Thus the attacker can execute SQL commands as shown below. Here we have used the payload below to extract the database name: **q=socks' or 1=1 UNION SELECT database(),database(),database(),database(),database(),database(),database()--+**



PoC – The attacker can dump arbitrary data

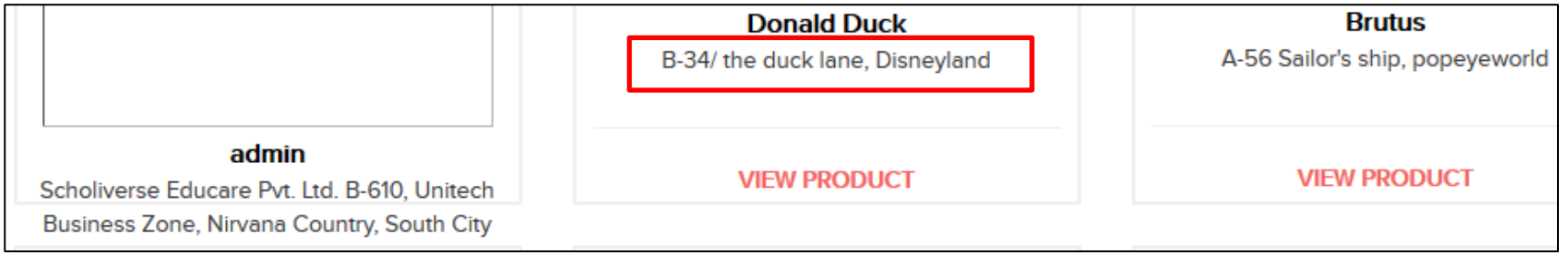
- No. of databases: 1
  - hacking_training_project

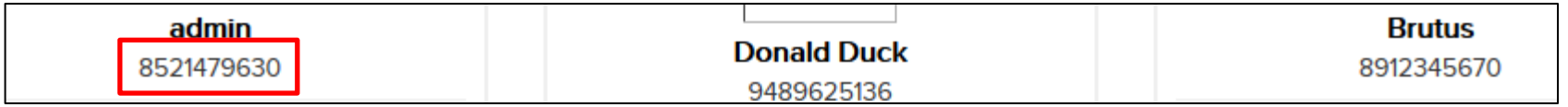
- No. of tables in hacking_training_project: 10
  - brands
  - cart_items
  - categories
  - customers
  - order_items
  - orders
  - product_reviews
  - products
  - sellers
  - users (may store some critical user information)

- No. of columns in users: 11 – id, type, unique_key, email, user_name, name, password, phone_number, address, created_at, last_updated_at

Business Impact – Extremely High

- Using this vulnerability, an attacker can execute arbitrary SQL commands on Lifestyle Store server and gain complete access to internal databases along with all customer data inside it.
- This data becomes critical since it leaks out the personal information like names, addresses, e-mail ids and phone numbers as shown below.
- The password leakage has been prevented by using proper encryption but the rest of the data is not encrypted and hence can be accessed.





Recommendations

Take the following precautions to avoid exploitation of SQL injections:

- Whitelist User Input: Whitelist all user input for expected data only. For example if you are expecting a flower name, limit it to alphabets only up to 20 characters in length. If you are expecting some ID, restrict it to numbers only.
- Prepared Statements: Use SQL prepared statements available in all web development languages and frameworks to avoid attacker being able to modify SQL query.
- Character encoding: If you are taking input that requires you to accept special characters, encode it. Example. Convert all **‘ to \’** , **“ to \”**, **\ to \\.** It is also suggested to follow a standard encoding for all special characters such has HTML encoding, URL encoding etc.
- Do not 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

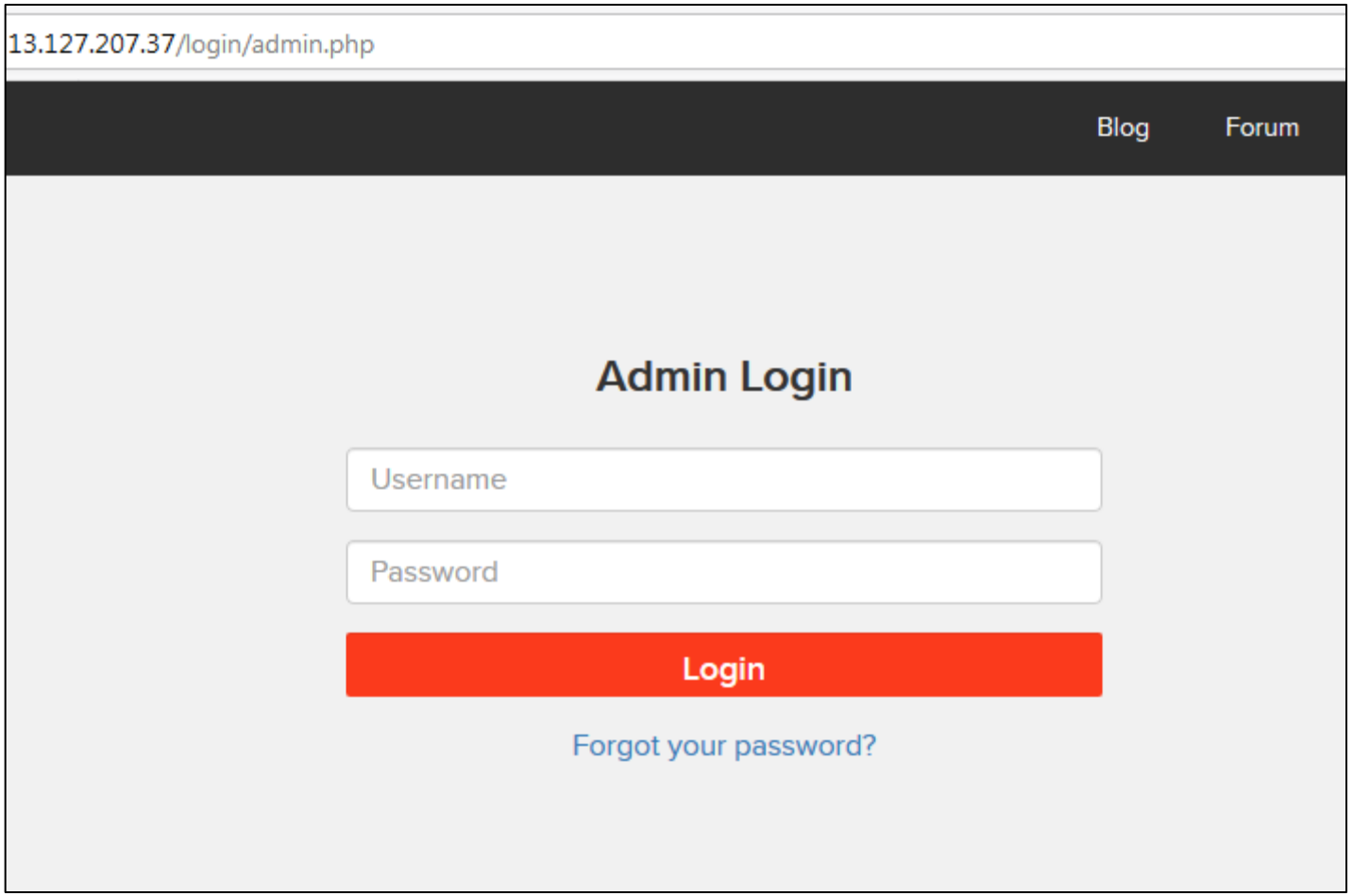
- *https://www.owasp.org/index.php/SQL_Injection*
- *https://en.wikipedia.org/wiki/SQL_injection*

2. Access to Admin Panel

Access to Admin dashboard panel and console (Critical)	Below mentioned URL in the **Lifestyle Store section** is vulnerable to default/weak passwords and url **Affected URL:** ➢ http://url/login/admin.php ➢ http://url/admin31/dashboard.php **Affected Parameters:** ➢ username, password (POST parameters) **Payload:** ➢ username = admin, password = admin

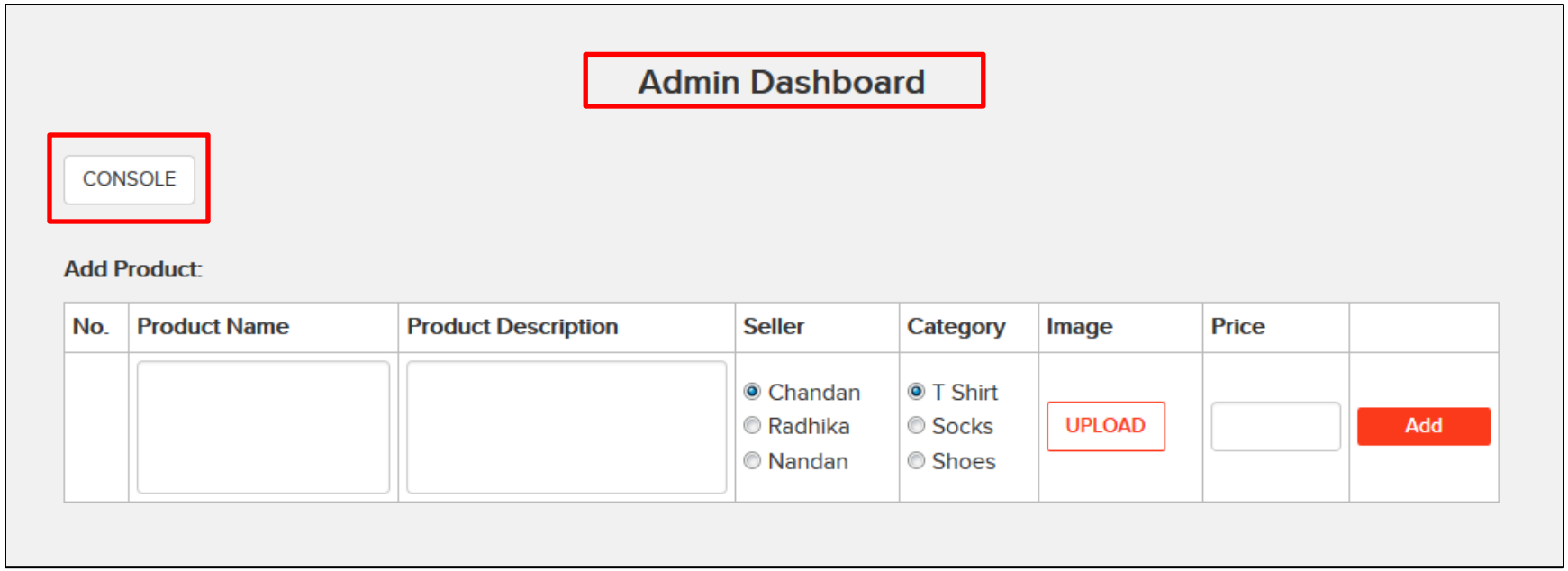
Observation

- Navigate to **http://url/login/admin.php** and you will see the login page



Observation

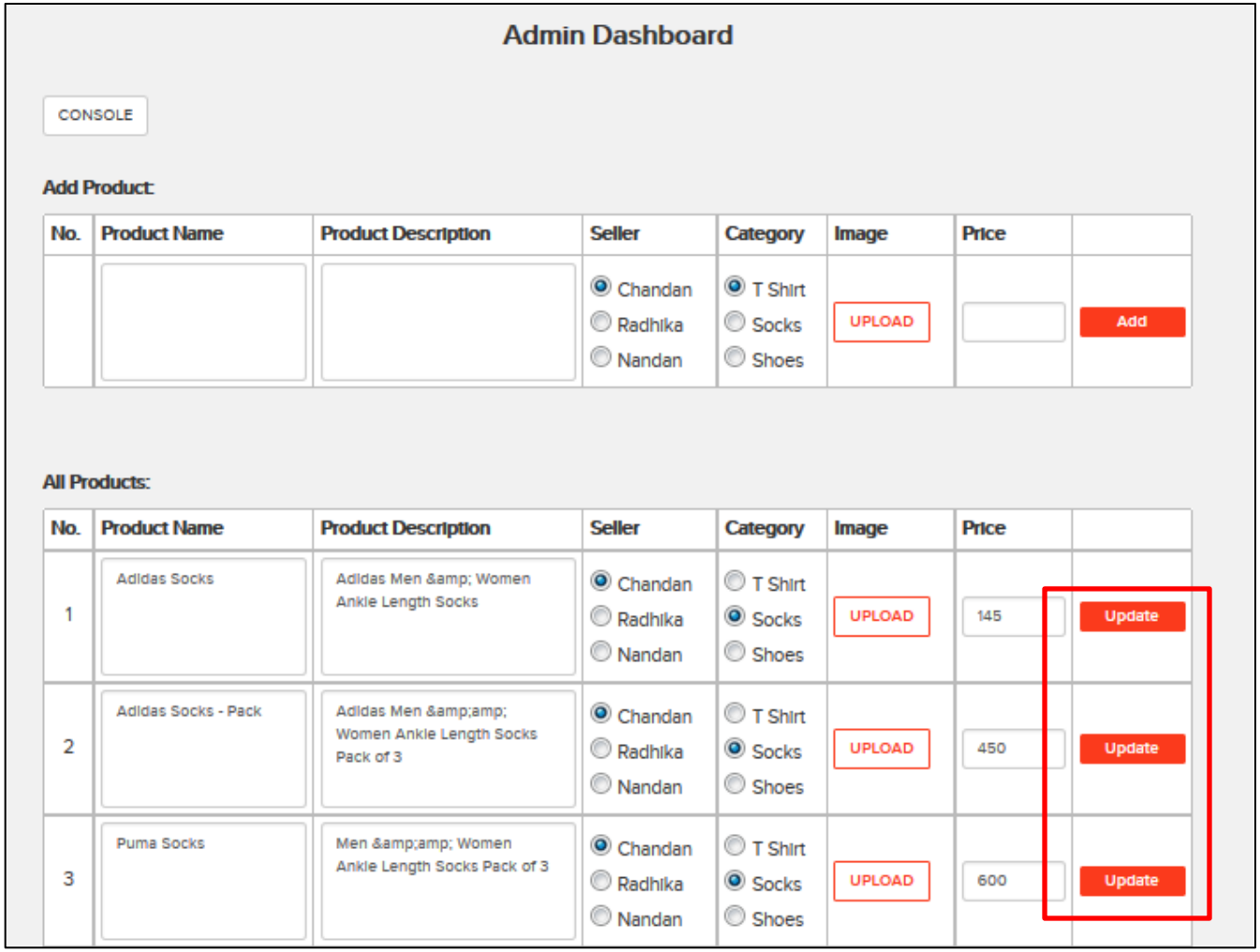
- Enter username: admin and password: admin. You will get logged into the admin panel and will be greeted by the admin dashboard.
- From here you can also access the console.



- Further more this admin dashboard can be accessed directly by any logged in user: **http://url/admin31/dashboard.php** which is a critical authorization flaw.

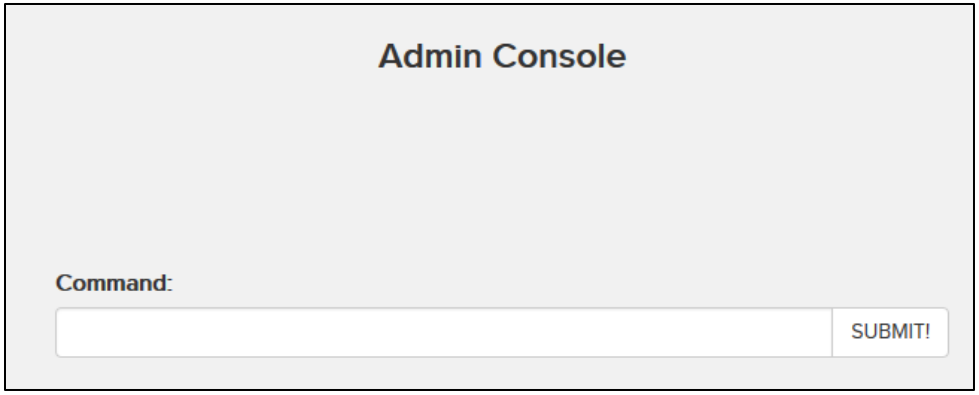
PoC

- The attacker can modify, add or delete any product details.

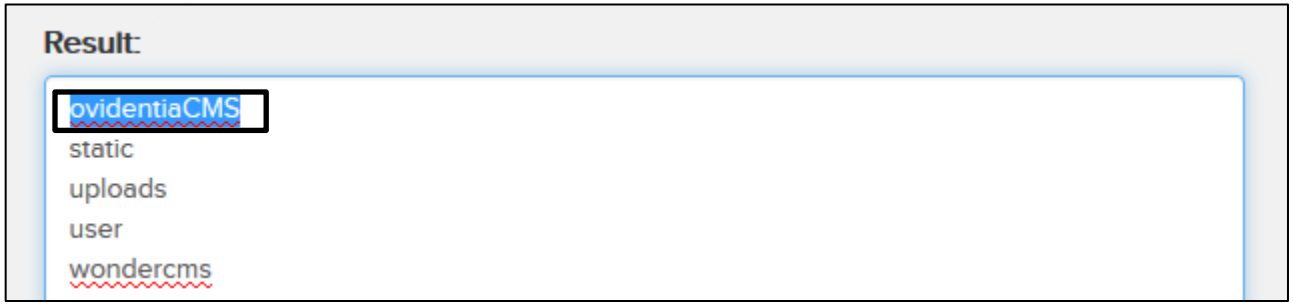


PoC

- The attacker can also access the console: **http://url/admin31/console.php**



- Using -ls command: here we find something called **ovidentiaCMS.** Which looks to be hidden from the users.



Business Impact – Extremely High

- A malicious hacker can access the admin dashboard which can to used to edit many critical information on the website such as:
  - ❑ Product Name
  - ❑ Product Description
  - ❑ Seller
  - ❑ Price of the product
  - ❑ Image of the product

- Furthermore the hacker can access the admin console which can be used to know more information about the sever on which the website is running or the directories that exist in the website.

Recommendations

Take the following precautions to avoid exploitation of default/weak passwords:

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

References

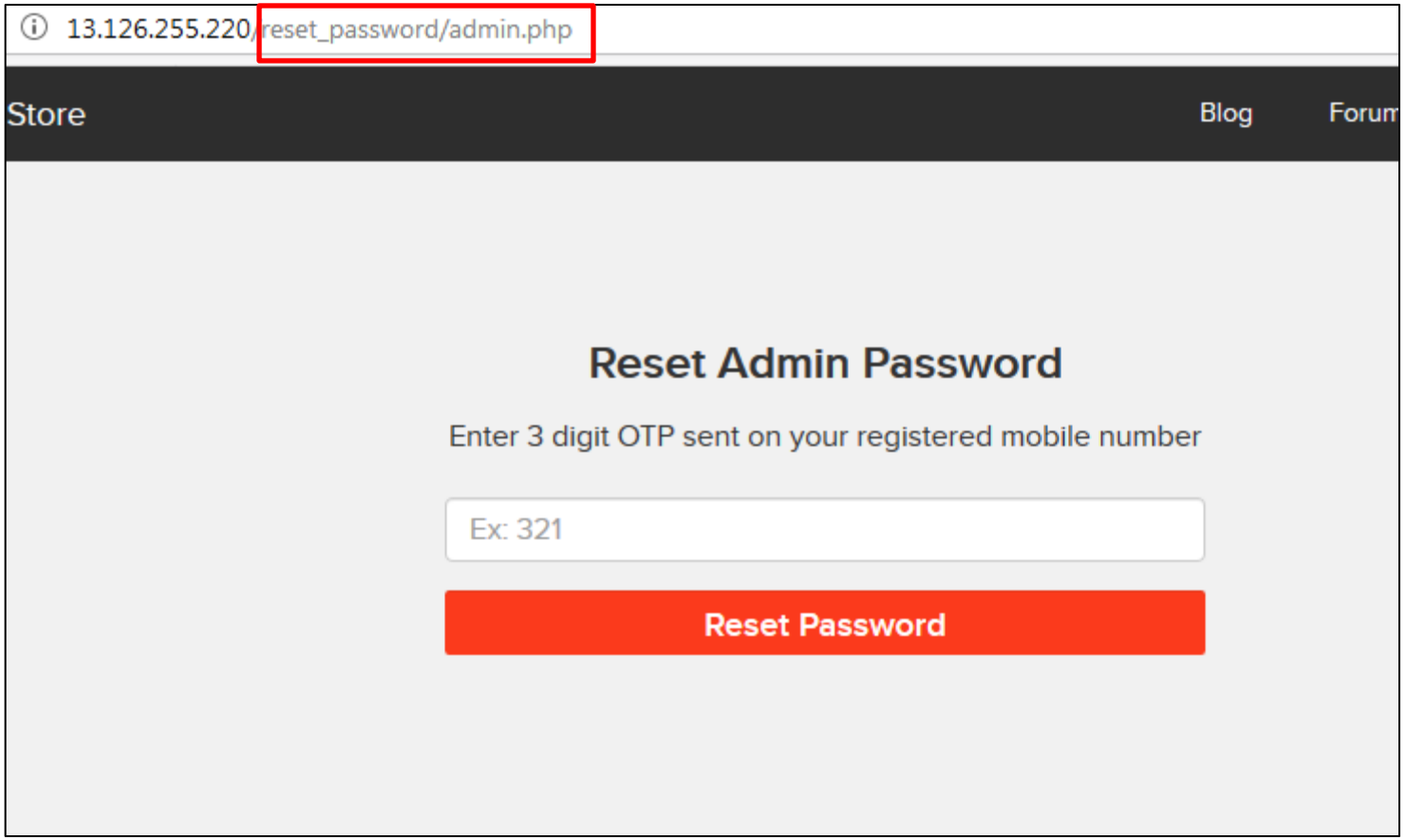
- *https://www.owasp.org/index.php/Testing_for_weak_password_change_or_reset_functionalities_(OTG-AUTHN-009)*
- *https://www.owasp.org/index.php/Default_Passwords*

3. Admin Account takeover via OTP bypass

Admin Account takeover via OTP bypass (Critical)	Below mentioned URL in the **Lifestyle Store section** is vulnerable to admin account take over using OTP bypass. This is due to Rate Limiting Flaws and brute forcing. **Affected URL:** ➢ http://url/reset_password/admin.php **Affected Parameter:** ➢ OTP (post parameter)

Observation

- Navigate to **http://url/reset_password/admin.php** and you will see the reset password page. Enter any random three digit number and click on reset password.



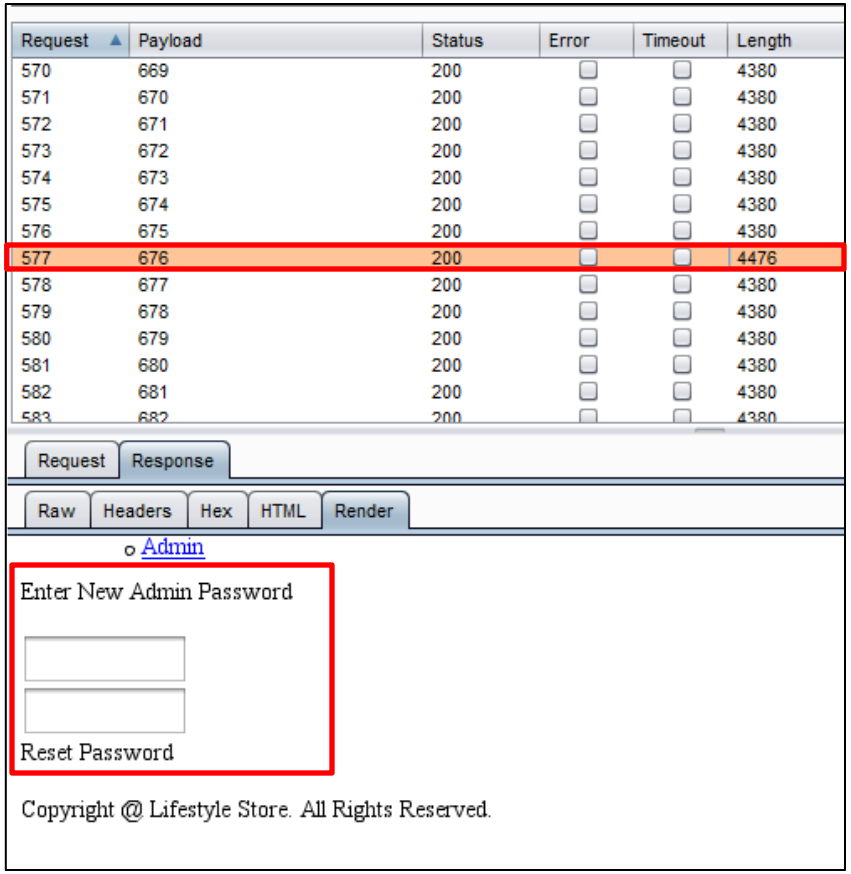
Observation

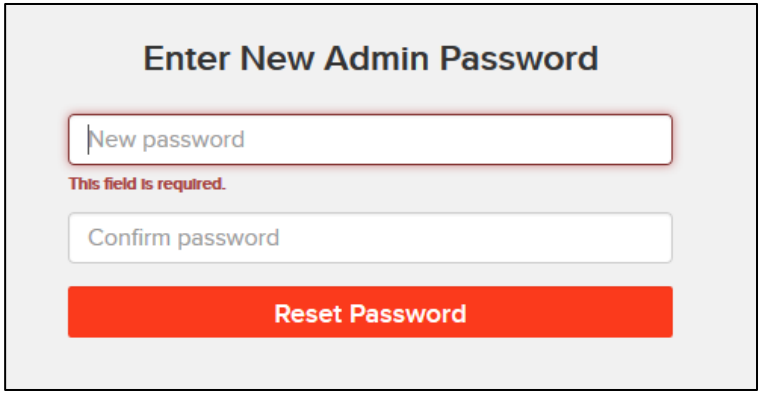
- The following request is generated containing the post OTP parameter.

```
GET /reset_password/admin.php?otp=$100$ HTTP/1.1
Host: 13.126.255.220
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:65.0) Gecko/20100101 Firefox/65.0
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
Referer: http://13.126.255.220/reset_password/admin.php
Connection: close
Cookie: key=9A84BDF0-310B-9020-BB2D-585A402E2D9D; PHPSESSID=dsqligbe39hgimso2s57fcva21;
Upgrade-Insecure-Requests: 1
```

Observation

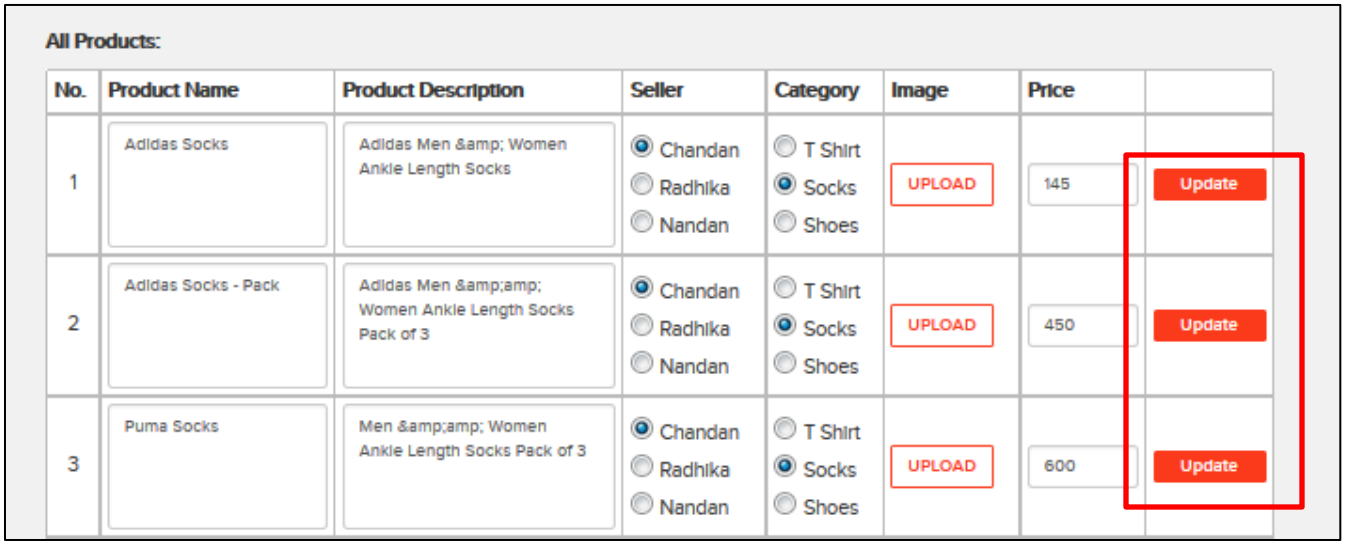
- We shoot the request with all possible combinations of 3 Digit OTPs and upon a successful hit, we get a response containing user details. We can use the same OTP then to login.

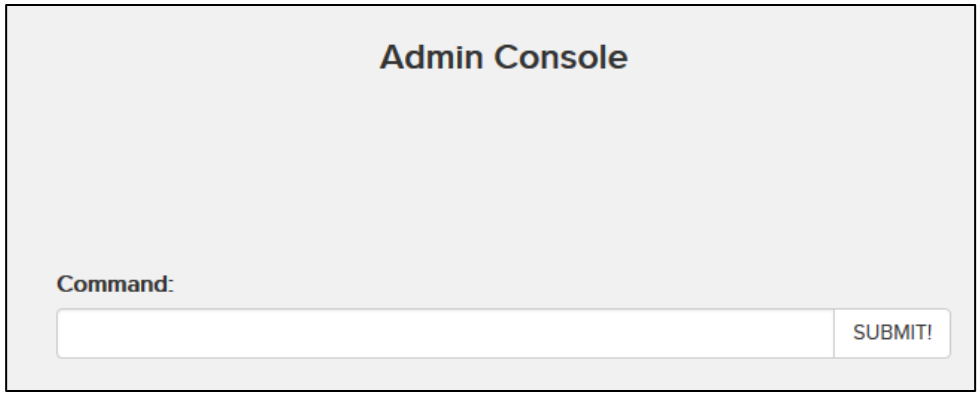




PoC

- The attacker can modify, add or delete any product details and access the console.





Business Impact – Extremely High

- A malicious hacker can access the admin dashboard which can to used to edit many critical information on the website such as:
  - ❑ Product Name
  - ❑ Product Description
  - ❑ Seller
  - ❑ Price of the product
  - ❑ Image of the product

- Furthermore the hacker can access the admin console which can be used to know more information about the sever on which the website is running or the directories that exist in the website.

Recommendations

Take the following precautions to avoid exploitation of OTP bypass:

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

4. Unauthorized access to customer details.

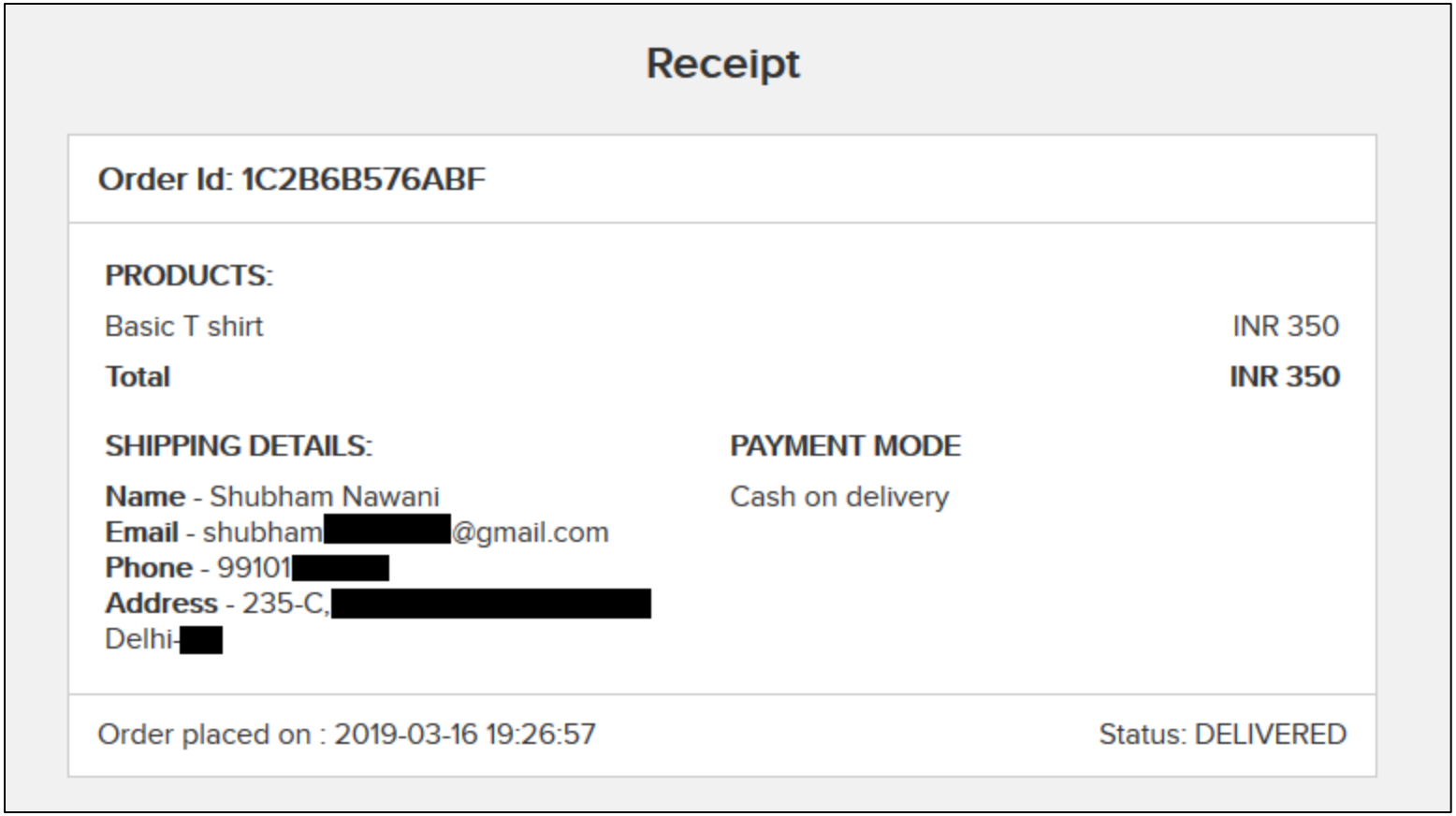
Unauthorized access to customer details via IDOR and Rate Limiting Flaws (Critical)	Below mentioned URL in the **Lifestyle Store section** is vulnerable to Unauthorized access to customer details via IDOR and Rate Limiting Flaws. **Affected URL:** ➢ http://url/orders/generate_receipt/ordered/6=HERE **Affected Parameter:** ➢ /6 (GET Parameter) **Payload:** ➢ 6

4. Unauthorized access to customer details.

Unauthorized access to customer details via IDOR and Rate Limiting Flaws (Critical)	Another URL in the website is affected to the same vulnerability. **Affected URL:** ➢ http://url/orders/orders.php?customer=13 **Affected Parameter:** ➢ customer=13 (GET Parameter) **Payload:** ➢ 13

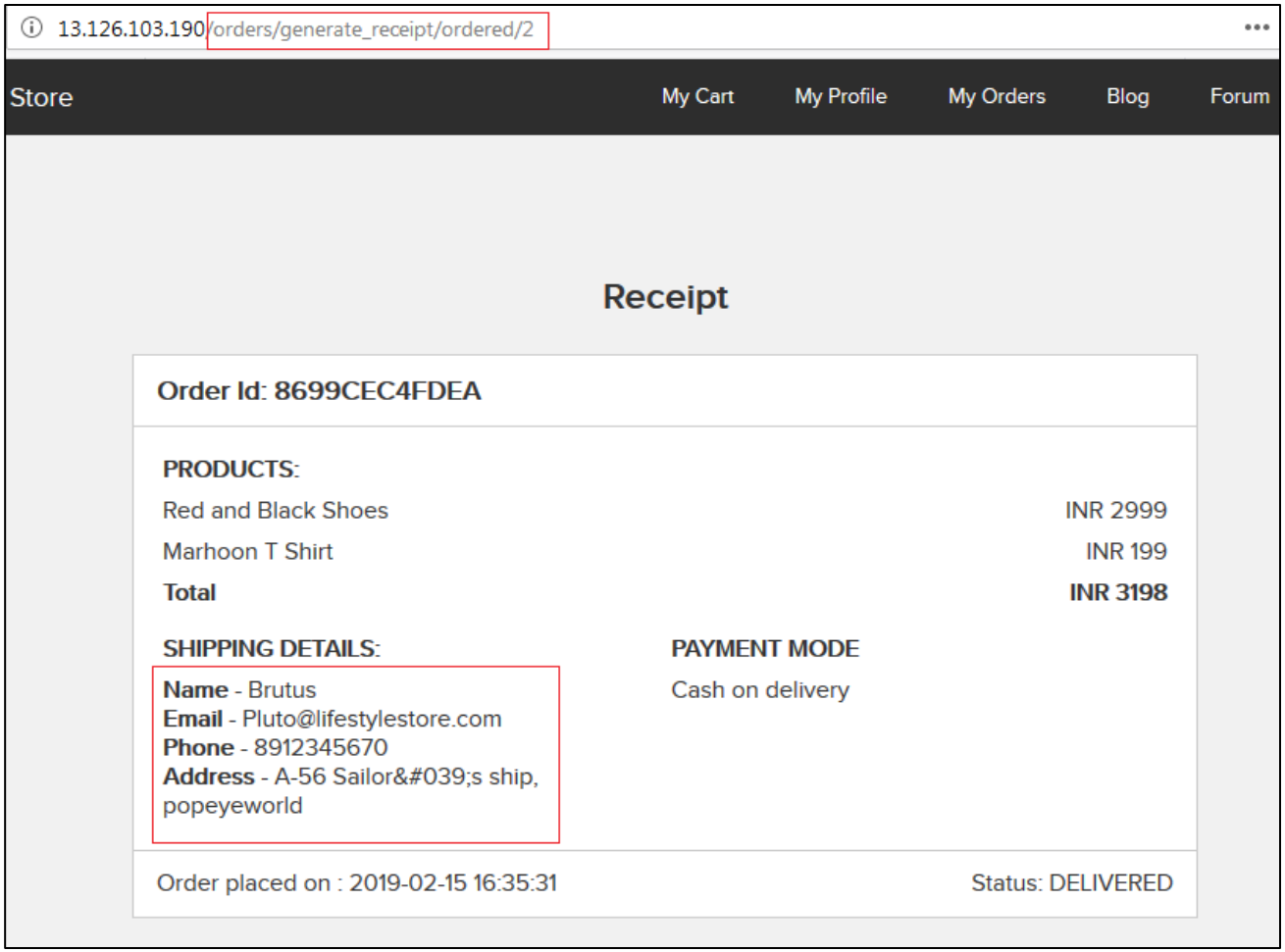
Observation

- Log in to your account and buy any product from the product list. After buying the item you will be redirected to: **http://url/orders/generate_receipt/ordered/6**
- Here notice the GET based parameter '6'.



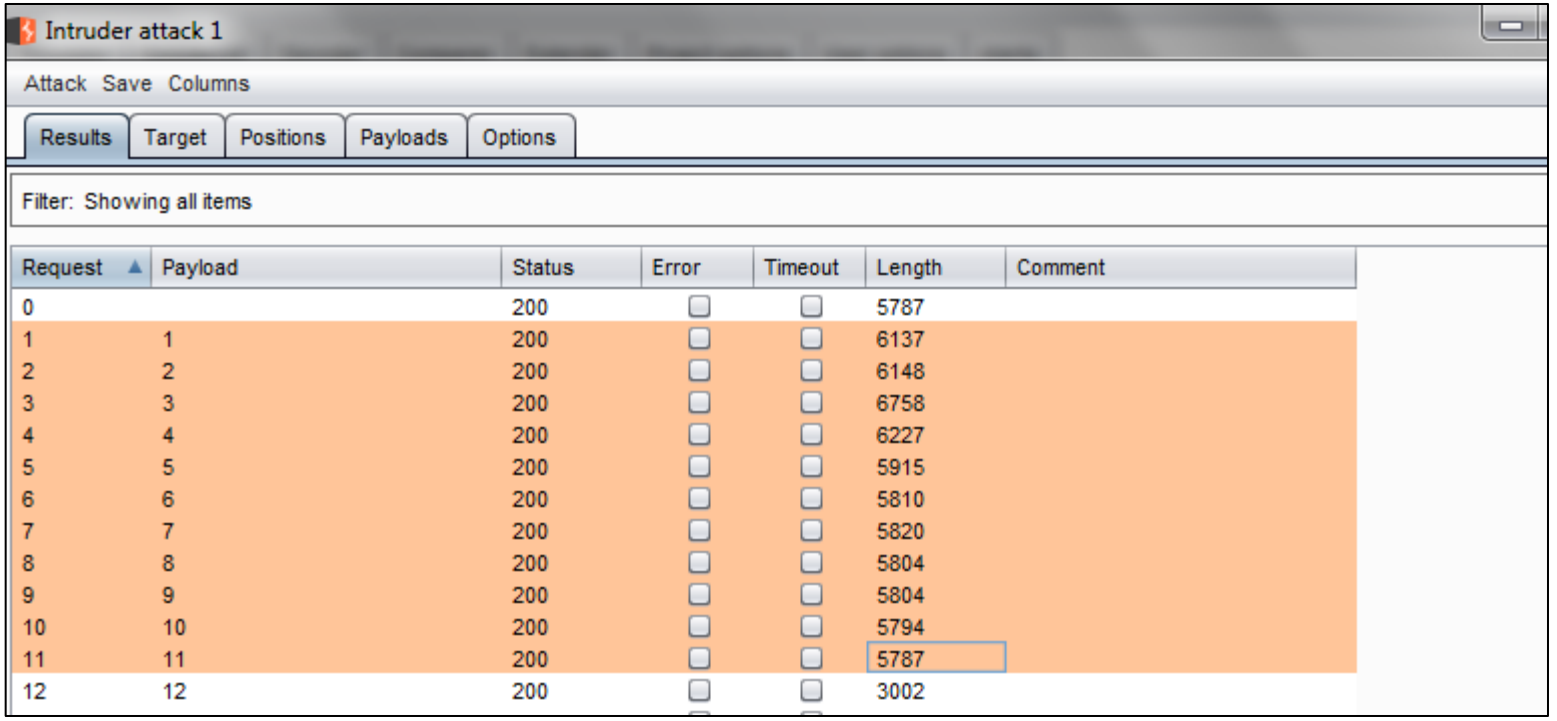
Observation

- Notice the url: **http://url/orders/generate_receipt/ordered/6** and change the parameter from ‘6’ to ‘2’



PoC

- The attacker can access all the generated receipts. All the below requests in the are accessible.



Request	Payload	Status	Error	Timeout	Length	Comment
0		200	☐	☐	5787	
1	1	200	☐	☐	6137	
2	2	200	☐	☐	6148	
3	3	200	☐	☐	6758	
4	4	200	☐	☐	6227	
5	5	200	☐	☐	5915	
6	6	200	☐	☐	5810	
7	7	200	☐	☐	5820	
8	8	200	☐	☐	5804	
9	9	200	☐	☐	5804	
10	10	200	☐	☐	5794	
11	11	200	☐	☐	5787	
12	12	200	☐	☐	3002	

Business Impact – Extremely High

- A malicious hacker can access the critical information including:
  - ❑ User Name
  - ❑ E-mail ID
  - ❑ Order ID
  - ❑ Phone number
  - ❑ Address

- This can be used by malicious hackers to carry out targeted phishing attacks on the users and the information can also be sold to competitors/black market.

- More over, as there is no rate limiting checks, attacker can brute force for all possible values and get personal information of each and every user of the organization resulting is a massive information leakage.

- Other IDORs on the website are leaking similar information.

Recommendations

Take the following precautions to avoid exploitation of IDOR and Rate limiting flaws:

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

References

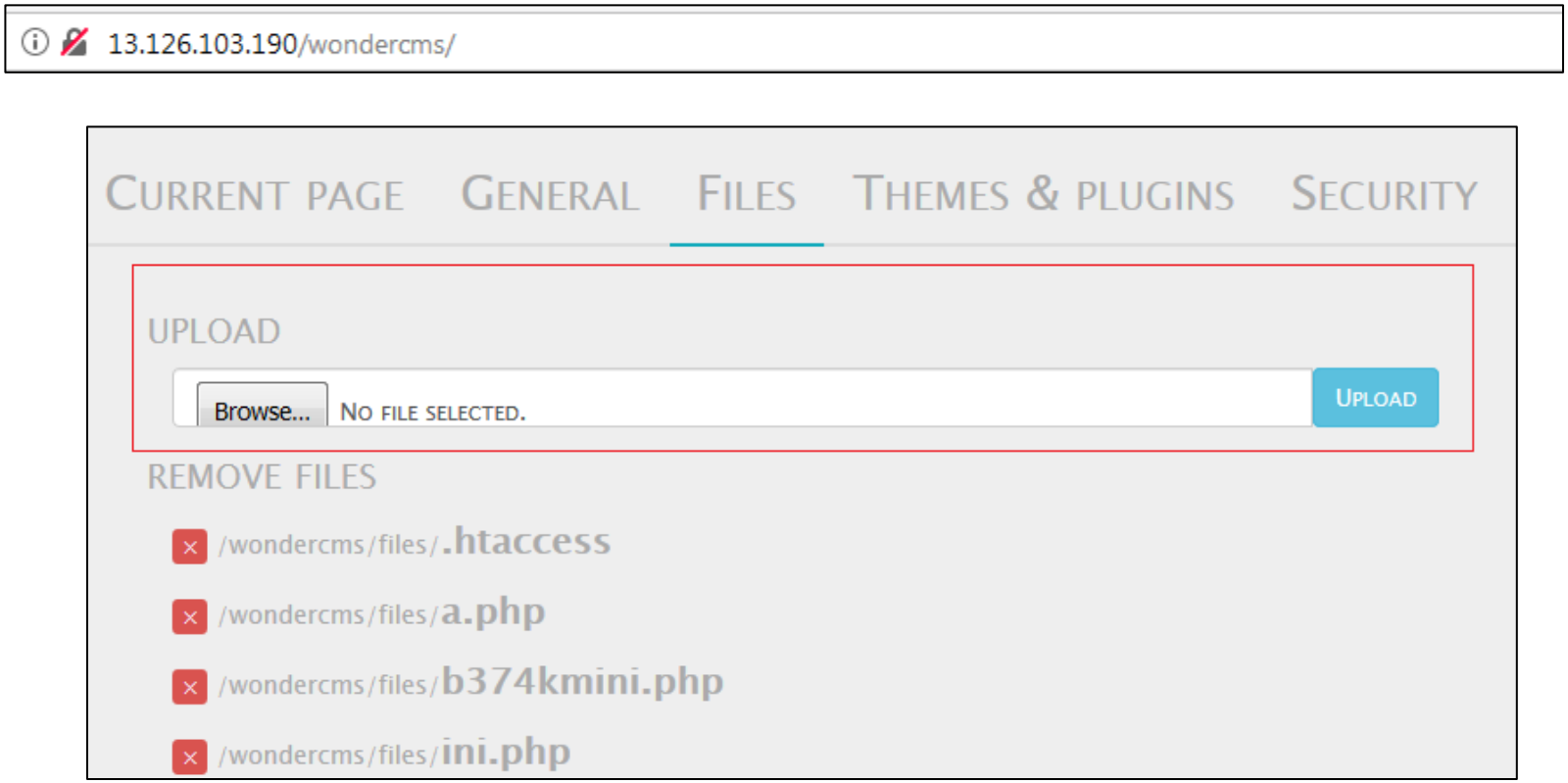
- *https://www.owasp.org/index.php/Insecure_Configuration_Management*
- *https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References*

5. Arbitrary File Upload

Arbitrary File Upload (Critical)	Below mentioned URL in the **WonderCMS section** is vulnerable to Arbitrary File Upload. **Affected URL:** ➢ http://url/wondercms/ **Payload:** ➢ Arbitrary File

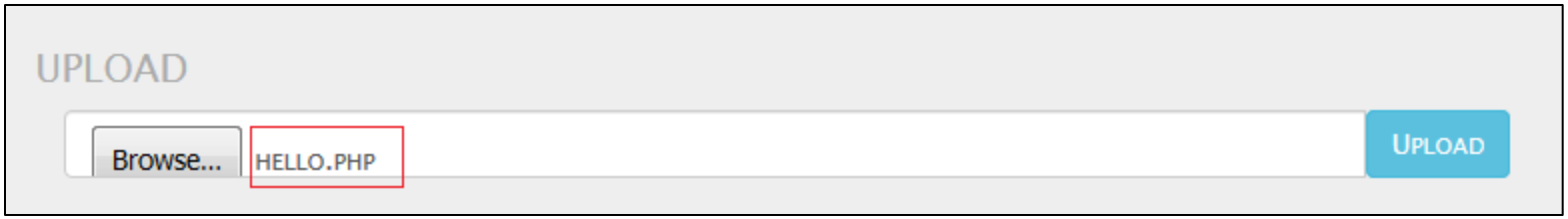
Observation

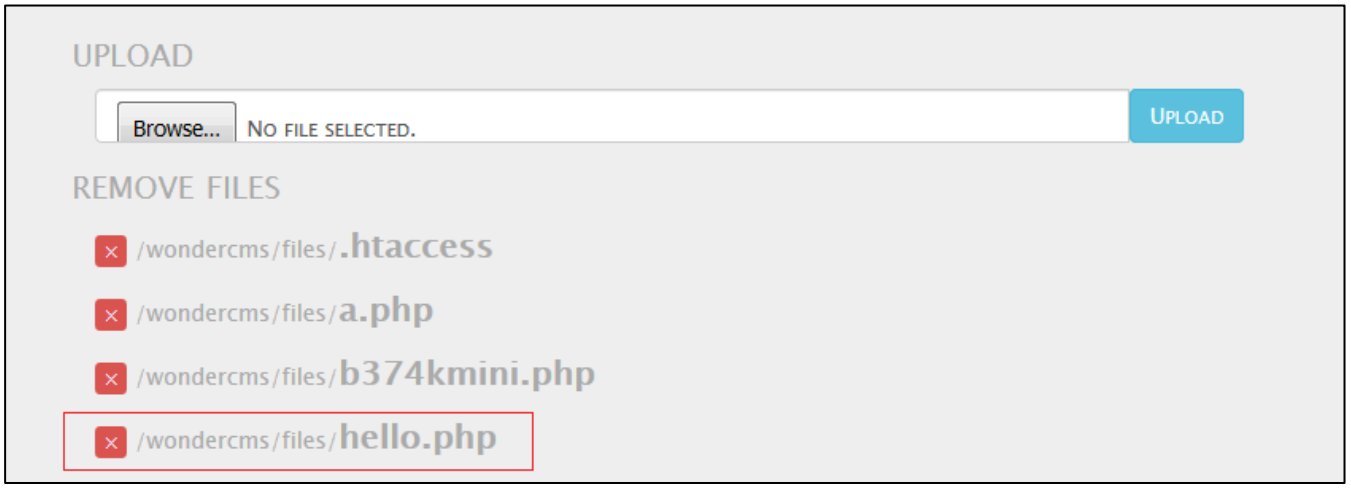
- Access the blog from the original website.
- Log in to your account using the default admin password and head to the setting tab and go for the files menu in: **http://url/wondercms/**

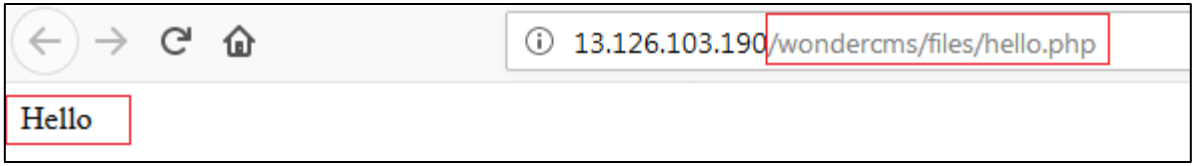


Observation

- Upload any arbitrary file: **hello.php**. After successfully uploading the file navigate back to file menu and access the file. "Hello world" will come out to be printed i.e. the code in the php file was executed.

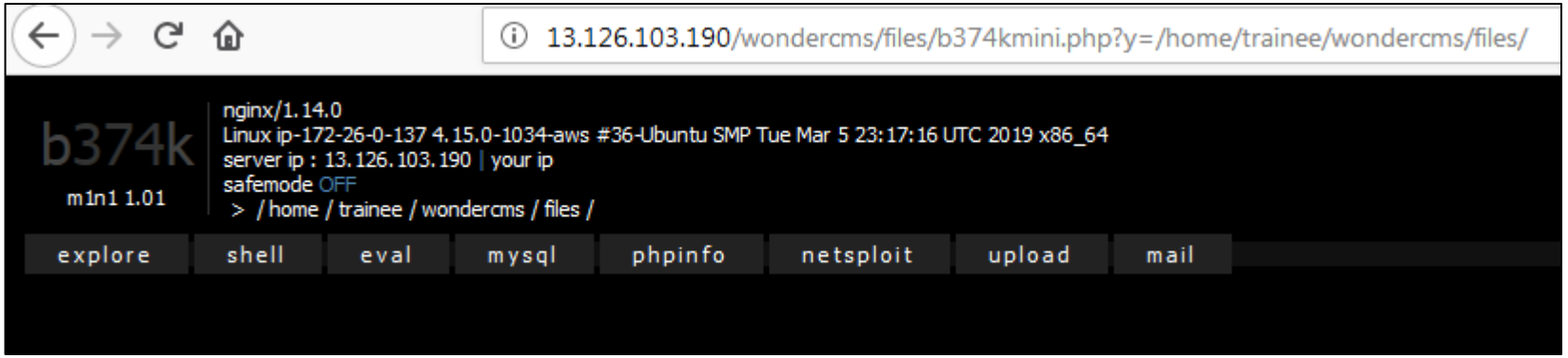


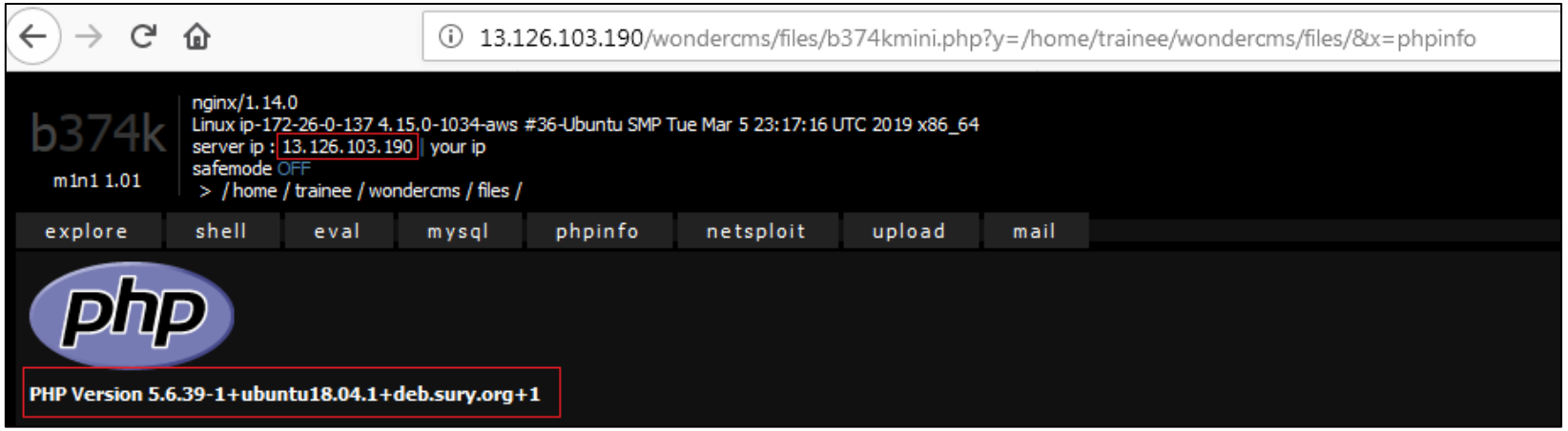




PoC

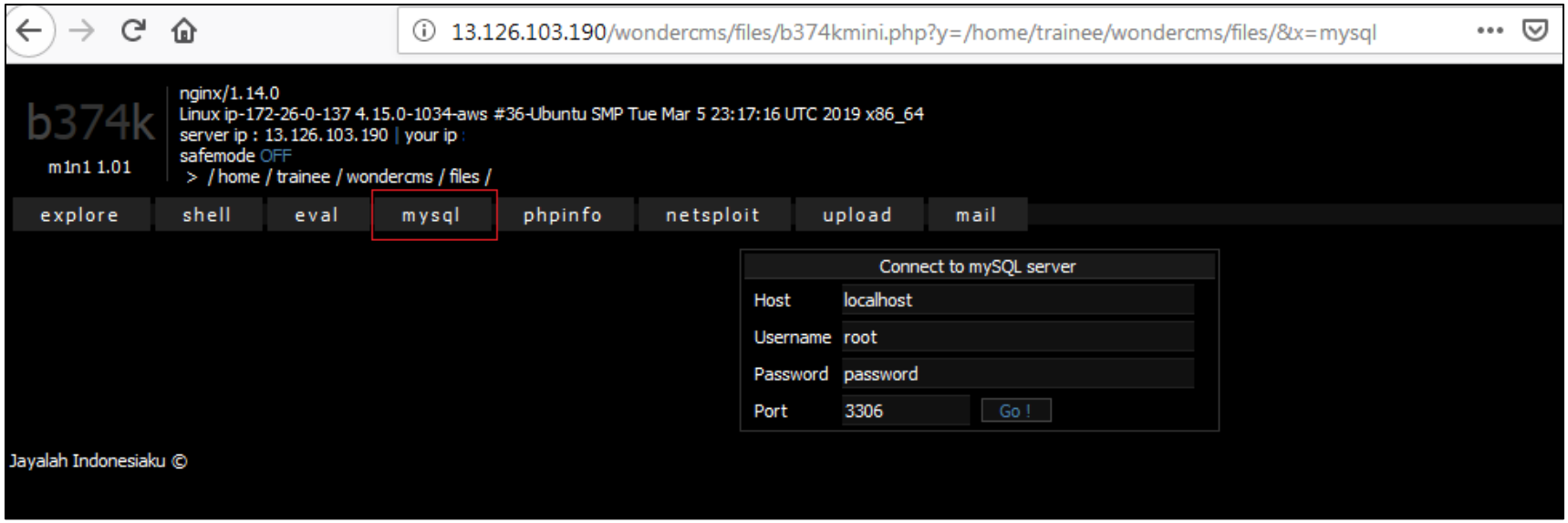
- The attacker can upload malicious scripts or shells and can access or take over the whole server. Knowing critical information stored in the server and information about the server.





PoC

- The attacker can access the database and edit, add or delete critical user information stored on the server.



Business Impact – Extremely High

- A malicious hacker upload malicious scripts or shells and can access or take over the whole server. Knowing critical information stored in the server and information about the server.
- The impact of this vulnerability is high, supposed code can be executed in the server context or on the client side. The likelihood of detection for the attacker is high. The prevalence is common. As a result the severity of this type of vulnerability is high
- A malicious file such as a Unix shell script, a windows virus, an Excel file with a dangerous formula, or a reverse shell can be uploaded on the server in order to execute code by an administrator or webmaster later -- on the victim's machine.
- An attacker might be able to put a phishing page into the website or deface the website.
- Uploaded sensitive files might be accessible by unauthorised people.
- File uploaders may disclose internal information such as server internal paths in their error messages.

Recommendations

Take the following precautions to avoid exploitation of Arbitrary File Upload:

- 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.
- The application should perform filtering and content checking on any files which are uploaded to the server. Files should be thoroughly scanned and validated before being made available to other users. If in doubt, the file should be discarded.

References

- *https://www.owasp.org/index.php/Unrestricted_File_Upload*
- *https://soroush.secproject.com/blog/2010/03/improve-file-uploaders%e2%80%99-protections-rev-1-0/*

6. Components with known vulnerabilities

Components with known vulnerabilities (Critical)	Below mentioned URLs in the **WonderCMS section** is vulnerable to Reflected Cross Site Scripting and Default admin password. **Affected URL:** ➢ http://url/wondercms/example **Payload:** ➢ <script>alert(1)</script> **Affected URL:** ➢ http://url/wondercms/ **Payload:** ➢ password = admin

6. Components with known vulnerabilities

Components with known vulnerabilities (Critical)	Below mentioned URL in the **OvidentiaCMS section** is vulnerable to Default Admin ID and Password. **Affected URL:** ➢ http://url/ovidentiaCMS/index.php?tg=login&cmd=authform&msg=Connexion&err=&restricted=1 **Payload:** ➢ nickname= admin@admin.bab ➢ password = 012345678

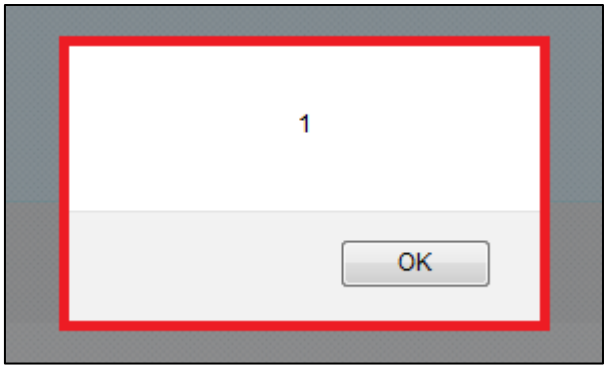
6. Components with known vulnerabilities

Components with known vulnerabilities (Critical)	Below mentioned URL in the **OvidentiaCMS section** is vulnerable to SQL Injection. **Affected URL:** ➢ http://url/ovidentiaCMS/index.php?tg=delegat&idx=mem&id=1 **Affected parameter:** ➢ id_dg **Payload:** ➢ id_dg = 1’

Observation and PoC - XSS

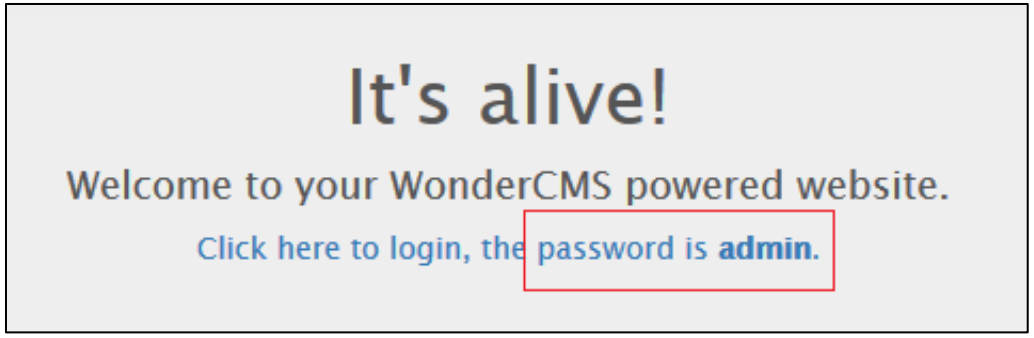
- Log in to the default admin account and navigate to : **http://wondercms/example**
  Now select the editable area and enter the code: **<script>alert(1)</script>**
- Now upon hitting enter, you will notice that the code has been executed, hence confirming XSS.

```
<h3>How to create new pages</h3>
<p><i>Settings -> General -> Add page</i></p>
<script>alert(1)</script>
<h3>How to edit anything</h3>
<p>Click anywhere inside the gray dashed area to edit. Click outside the area to save.</p>

<h3>How to install/update themes and plugins</h3>
<p>1. Copy link/URL to ZIP file.</p>
<p>2. Paste link in <i>Settings -> Themes and plugins</i> and click <i>Install/update</i>.</p>
<p><a href="https://github.com/robiso/wondercms-themes#list-of-approved-themes" target="_blank">WonderCMS themes</a> | <a
href="https://github.com/robiso/wondercms-plugins#approved-plugins" target="_blank">WonderCMS plugins</a></p>
```



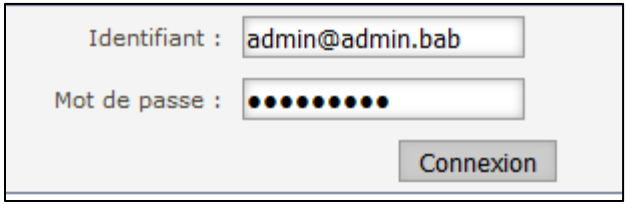
Observation and PoC – Default password and admin login URL

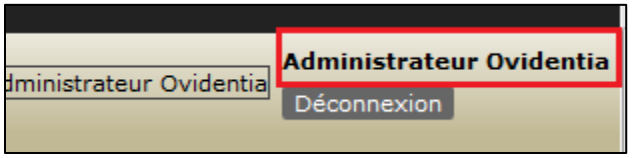
- In the wonderCMS section, you can log into the admin account by simply going to the url: **http://url/wondercms/loginURL** and using the password: **admin**
- Upon login you can edit all settings.



Observation and PoC – Default User ID and password

- In the OvidentiaCMS section, you can log into the admin account by simply going to the url: **http://urlovidentiaCMS/index.php?tg=login&cmd=authform&msg=Connexion&err=&restricted=1** and using the credentials:
- nickname: **admin@admin.bab** password**: 012345678**
- Upon login you get the admin privileges.

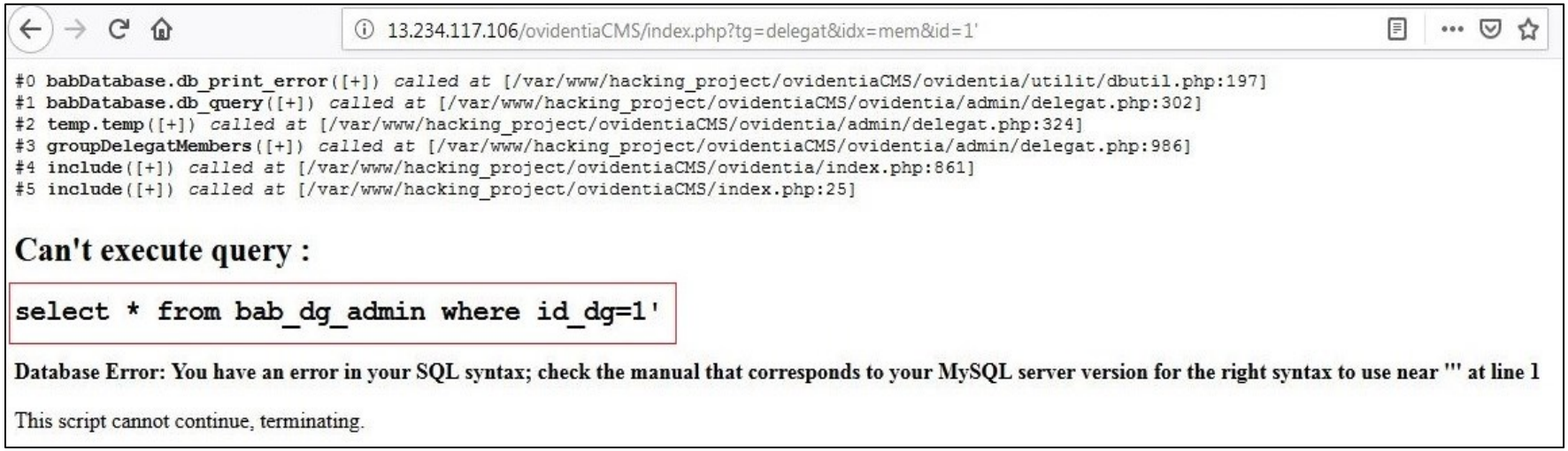





```
POST /ovidentiaCMS/index.php HTTP/1.1
Host: 13.234.117.106
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://13.234.117.106/ovidentiaCMS/index.php?tg=login&cmd=authform&msg=Connexion&err=&rest
ricted=1
Content-Type: application/x-www-form-urlencoded
Content-Length: 163
Connection: close
Cookie: key=9A84BDF0-310B-9020-BB2D-585A402E2D9D; PHPSESSID=9m4b9gj5pk98hljtgul0u6kpb1;
X-XSRF-TOKEN=b177cc310462fa8c83780095dedc81948ea03f6be616cd28f9efe849d9a502c0;
OV1623946714=6t8s65di4t35acqa82hhvlvh33
Upgrade-Insecure-Requests: 1

babCsrfProtect=bab5c98cc2e8f0349.72330107&tg=login&referer=index.php&login=login&sAuthType
=Ovidentia&nickname=admin%40admin.bab&password=012345678&submit=Connexion
```

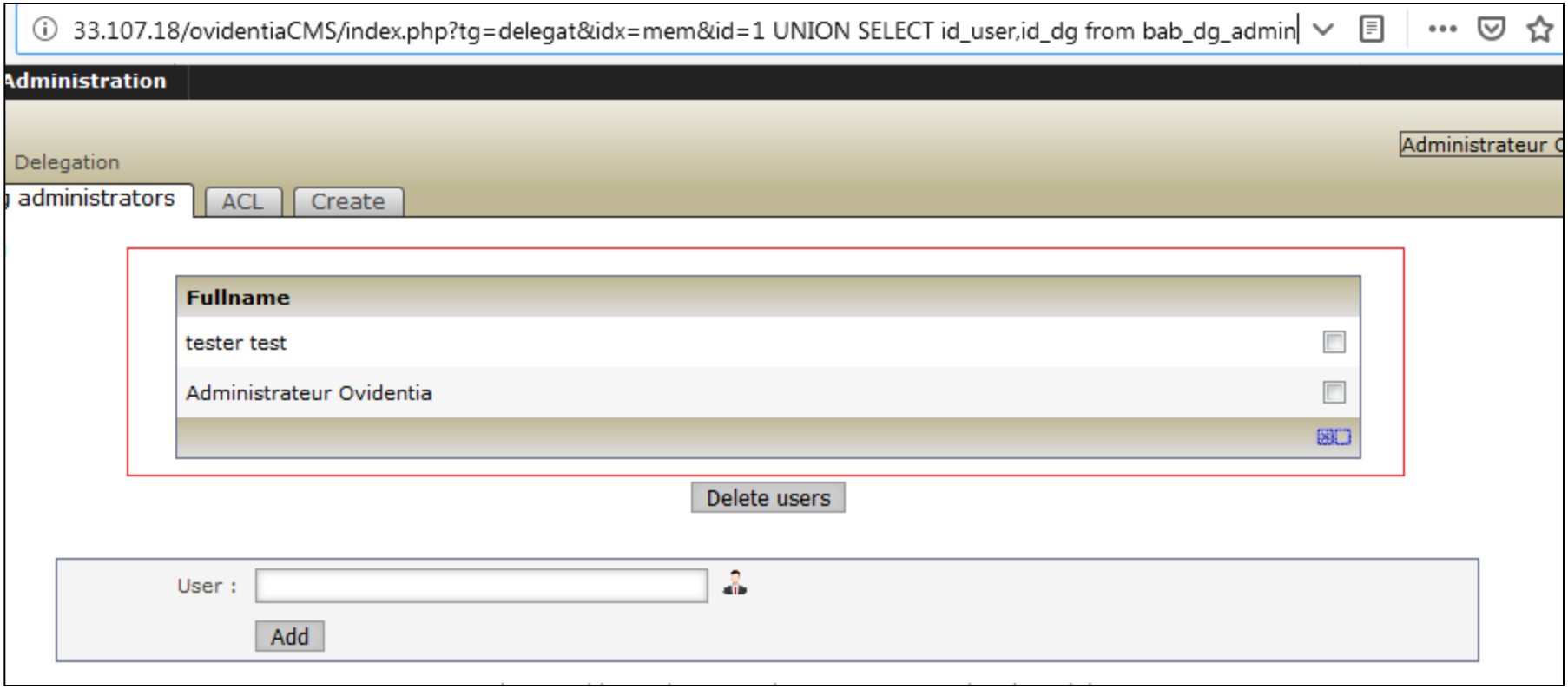
Observation – SQLi

- In the OvidentiaCMS section, after logging in as admin go to: **http://url/ovidentiaCMS/index.php?tg=delegat&idx=mem&id=1**
- Here notice the get parameter id, and then try out the following: **http://url/ovidentiaCMS/index.php?tg=delegat&idx=mem&id=1'** which throws an error.
- Hence the parameter is exploitable to SQLi.



PoC – SQLi

- Type the following to dump critical data:
  http://url/ovidentiaCMS/index.php?tg=delegat&idx=mem&id=1%20UNION%20SELECT%20id_user,id_dg%20from%20bab_dg_admin



Business Impact – Extremely High

- As attacker can inject arbitrary HTML CSS and JS via the URL, attacker can put any content on the page like phishing pages, install malware on victim's device and even host explicit content that could compromise the reputation of the organization.
- Using SQLi vulnerability, an attacker can execute arbitrary SQL commands on the OvidentiaCMS server and gain complete access to internal databases along with all user data inside it.
- The hacker can access the whole website using the default credentials and steal critical and vulnerable data.

Recommendations

Take the following precautions to avoid exploitation:

- Remove unused dependencies, unnecessary features, components, files, and documentation.
- Continuously inventory the versions of both client-side and server-side components (e.g. frameworks, libraries) and their dependencies using tools like versions ,DependencyCheck, retire.js, etc. Continuously monitor sources like CVE and NVD for vulnerabilities in the components. Use software composition analysis tools to automate the process.
- Subscribe to email alerts for security vulnerabilities related to components you use.
- Only obtain components from official sources over secure links.

References

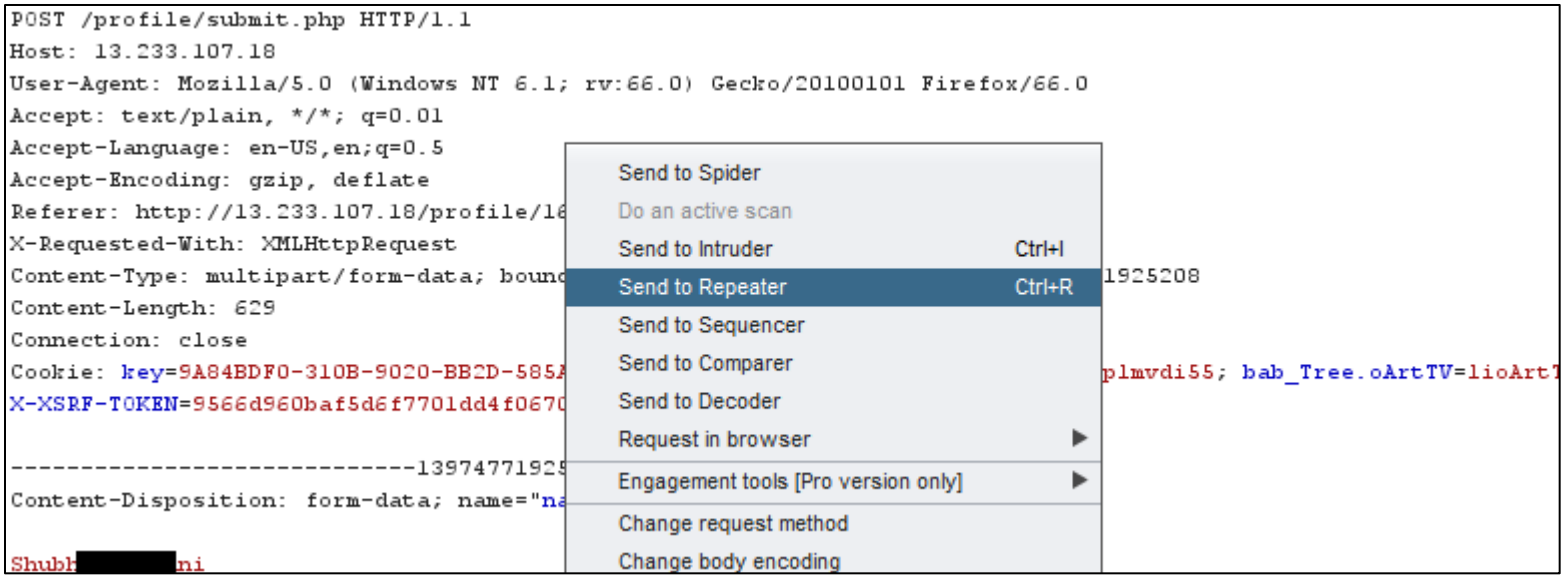
- *https://www.owasp.org/index.php/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities*
- *https://www.exploit-db.com/exploits/30107*

7. Cross Site Request Forgery

Cross Site Request Forgery (Critical)	Below mentioned URL in the **Life Style Store Section** is vulnerable to Cross Site Request Forgery **Affected URL:** ➢ http://url/profile/HERE/edit/ **Affected Parameter:** ➢ /profile/submit.php (POST parameter) **Payload:** ➢ name

Observation

- Log into your account and navigate to: **http://url/profile/HERE/edit/**
- Now turn the intercept ON and Fill in the details. Submit the request and see it in burp, send the request to repeater.



Observation

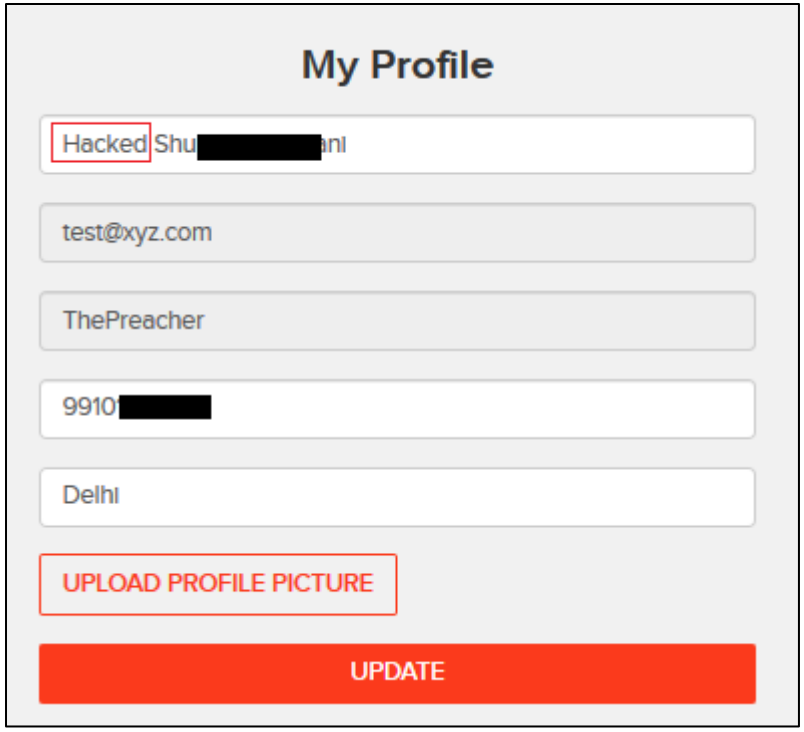
- Now make changes in the name parameter to: **Hacked** “name” and change the referrer header to **http://hacker.com**

```
POST /profile/submit.php HTTP/1.1
Host: 13.233.107.18
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/plain, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.hacker.com
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------13974771925208
Content-Length: 629
Connection: close
Cookie: key=9A84BDF0-310B-9020-BB2D-585A402E2D9D; PHPSESSID=pfl7picrlunro3i3okplmvdi55;
bab_Tree.oArtTV=lioArtTV.c__0/lioArtTV.c__1; 0V2933647928=l3o2f4n5qh8hptn7ck933pero6;
X-XSRF-TOKEN=9566d960baf5d6f7701dd4f067040bd406d62f579590974f227389cfa3b81cc4

-----------------------------13974771925208
Content-Disposition: form-data; name="name"

Hacked Shu[redacted]i
-----------------------------13974771925208
Content-Disposition: form-data; name="contact"

991019[redacted]
-----------------------------13974771925208
Content-Disposition: form-data; name="address"

Delhi
-----------------------------13974771925208
Content-Disposition: form-data; name="user_id"
```

Observation

- Now send the request you will notice that the request has been successfully accepted and changes are now made in the original website.
- Thus CSRF has been confirmed.

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 20 Mar 2019 15:14:04 GMT
Content-Type: text/html; charset=utf-8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
Pragma: no-cache
X-FRAME-OPTIONS: DENY
Set-Cookie:
X-XSRF-TOKEN=18edc3d360e591fafb300ae1a0709fca2d7e07312d4040fc01ddf
expires=Wed, 20-Mar-2019 16:14:04 GMT; Max-Age=3600; path=/
Content-Length: 64
Connection: close

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



Business Impact – Extremely High

- CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request.
- With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing.
- If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth.
- If the victim is an administrative account, CSRF can compromise the entire web application.

Recommendations

Take the following precautions to avoid exploitation of CSRF:

- 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.
- 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

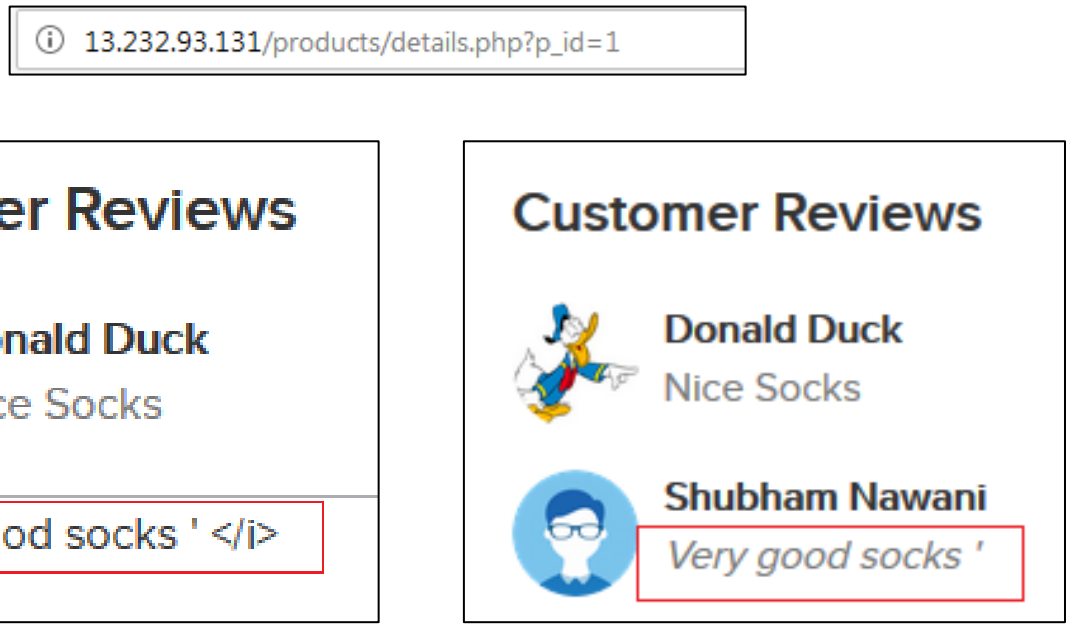
- *https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)*

8. Reflected Cross Site Scripting

Reflected Cross Site Scripting (Severe)	Below mentioned URL in the **Life Style Store section** is vulnerable to Reflected Cross Site Scripting. **Affected URL:** ➢ http://url/products/details.php?p_id=HERE **Affected Parameter:** ➢ p_id (GET Parameter) **Payload:** ➢ <script>alert(1)</script>

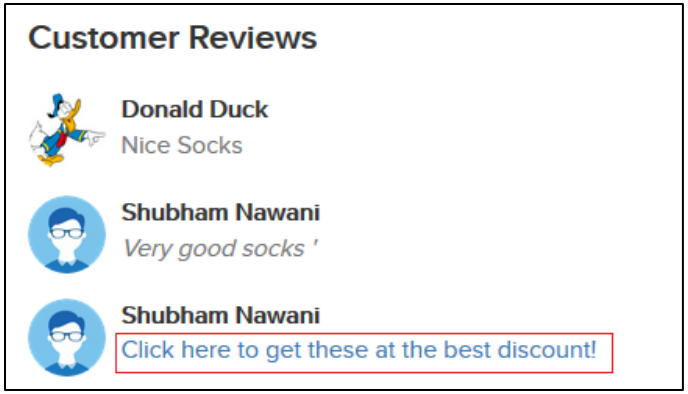
Observation

- Log in to your account and navigate to : **http://url/products.php**
- Now select any desired product and notice the comments section, try putting some html code in it.
- Now upon hitting enter, you will notice that the html code has been executed.



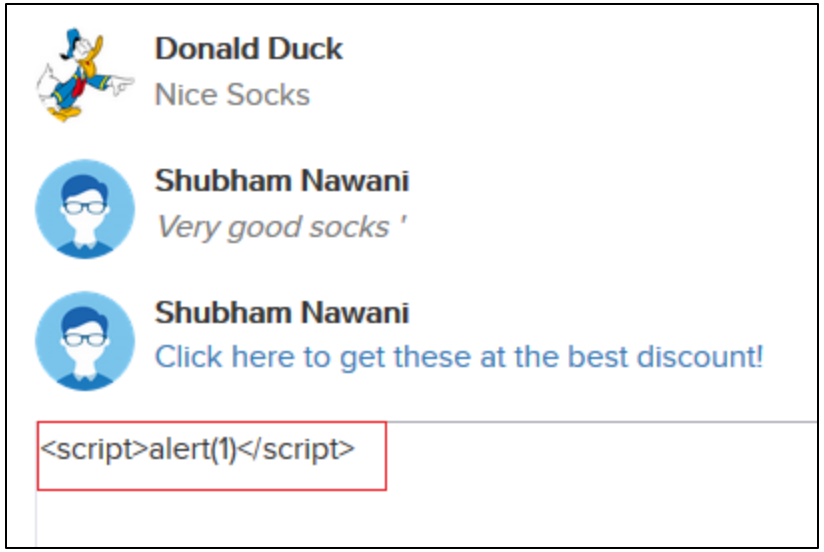
Observation

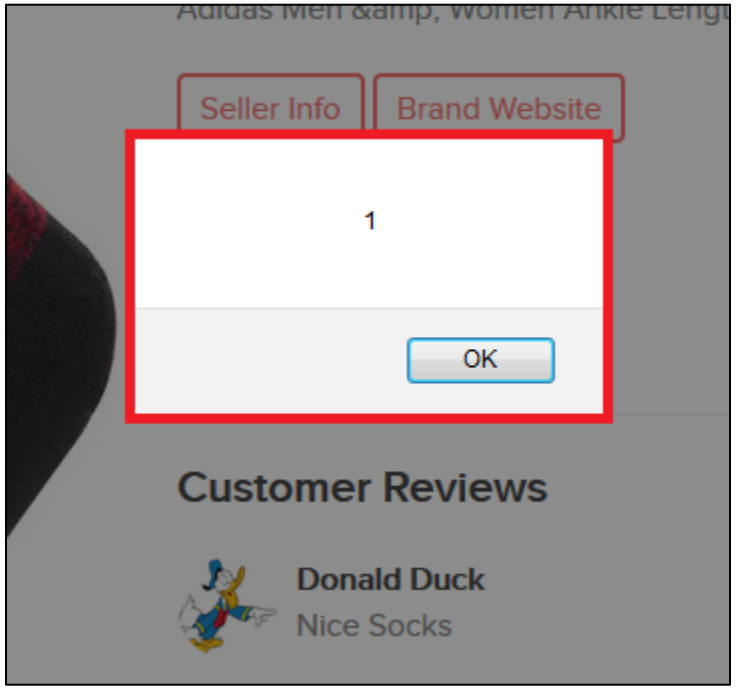
- Now use the following as payload: ** Click here to get these at the best discount!** hit enter and observe, a new hyperlink has appeared, which redirects the user to http://hacker.com
- Right click and inspect element to see that the code is now included in the HTML code.



PoC

- Now put the payload as: **<script>alert(1)</script>** in the comment box and hit enter, you will be presented with a dialog box. Which confirms Reflected XSS.





Business Impact – High

- As attacker can inject arbitrary HTML CSS and JS via the URL, attacker can put any content on the page like phishing pages, install malware on victim's device and even host explicit content that could compromise the reputation of the organization,
- All attacker needs to do is send the link with the payload to the victim and victim would see hacker controlled content on the website. As the user trusts the website, he/she will trust the content.

Recommendations

Take the following precautions to avoid exploitation of Reflected Cross Site Scripting:

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

References

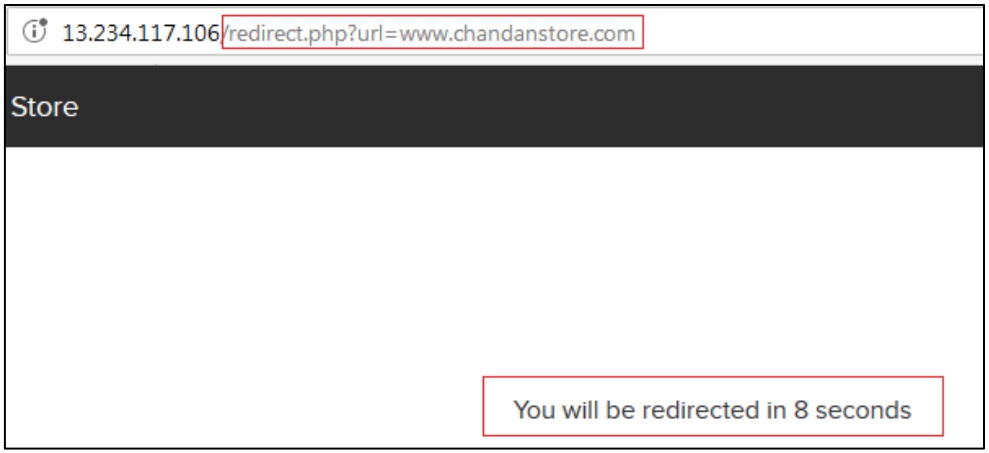
- *https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)*
- *https://en.wikipedia.org/wiki/Cross-site_scripting*
- *https://www.w3schools.com/html/html_entities.asp*

9. Open Redirection

Open Redirection (Severe)	Below mentioned URL in the **Life Style Store section** is vulnerable to Open Redirection. **Affected URL:** ➢ http://url/redirect.php?url=www.chandanstore.com **Affected Parameter:** ➢ url (referrer) **Payload:** ➢ www.google.com

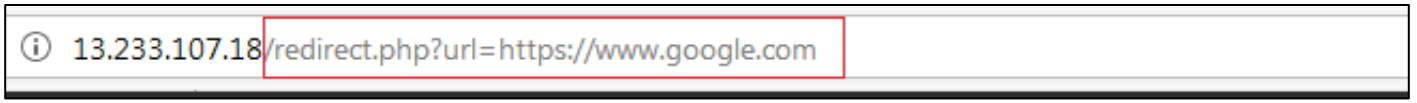
Observation

- Log in to your account and navigate to : **http://url/products.php**
- Now select any desired product and click on ‘Brand Website’ option.
- You will be redirected to another website in the next 10 seconds. Look at the url parameter in the request: **http://url/redirect.php?url=www.chandanstore.com**



Observation

- Now change the redirect url to: **http://url/redirect.php?url=www.google.com** and check the request in burp suite.
- The referral header would be changed, click forward request and you would be redirected to www.google.com




```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://13.233.107.18/redirect.php?url=https://www.google.com
Connection: close
Cookie: 1P_JAR=2019-03-20-14;
NID=164=VRqiS3HOtWhqxmYgdnD__9gr0Vh9gsQAFXlbD358vM58xBg_4wHq4A5URLbEjngUaFwdjS3wD4GpCxAA8G
jG5kw0zHDgV-uXlVyzSbVJf4ItqbWGhzUlid04-sP_Y_3333XUXCbQvpfDNXSWIy0v4dWotXU77UzlfmmyHzVozfg;
 OGP=-5061451:; ANID=AHWqTUkN5PqQ29GI96XXy-5uC0pMeNLnM5TIUm5NRGrF0qi9uu02qKwwm5SNEQX3
Upgrade-Insecure-Requests: 1
```

Business Impact – High

- An attacker can construct a URL within the application that causes a redirection to an arbitrary external domain.
- This behaviour can be leveraged to facilitate **phishing** attacks against users of the application.
- The ability to use an authentic application URL, targeting the correct domain and with a valid SSL certificate (if SSL is used), lends credibility to the phishing attack because many users, even if they verify these features, will not notice the subsequent redirection to a different domain.
- As mentioned above the impacts can be many, and vary from theft of information and credentials, to the redirection to malicious websites containing attacker controlled content, which in some cases even cause XSS attacks. So even though an open redirection might sound harmless at first, the impacts of it can be severe should it be exploitable.

Recommendations

Take the following precautions to avoid exploitation of Open Redirection:

- Remove the redirection function from the application, and replace links to it with direct links to the relevant target URLs.
- 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.
- The easiest and most effective way to prevent vulnerable open redirects would be to not let the user control where your page redirects him to.

References

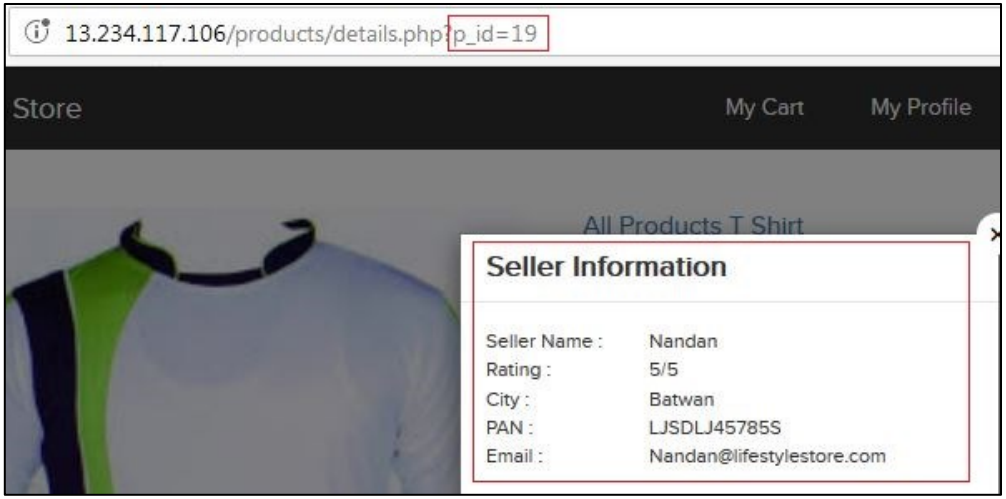
- *https://www.netsparker.com/blog/web-security/open-redirection-vulnerability-information-prevention*
- *https://portswigger.net/kb/issues/00500100_open-redirection-reflected*

10. PII Leakage and Forced Browsing

PII Leakage and Forced Browsing (Severe)	Below mentioned URL in the **Life Style Store section** is vulnerable to PII Leakage and Forced Browsing. **Affected URL:** ➢ http://url/products/details.php?p_id=HERE **Affected Parameter:** ➢ p_id **Payload:** ➢ p_id= HERE

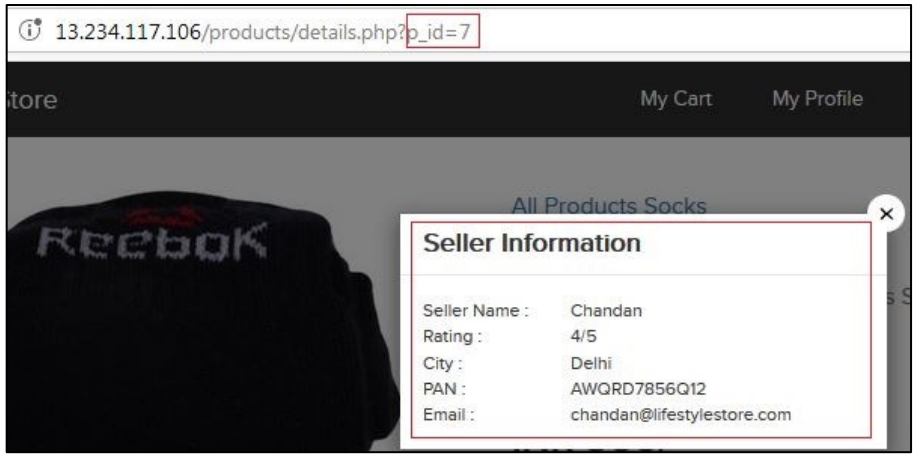
Observation

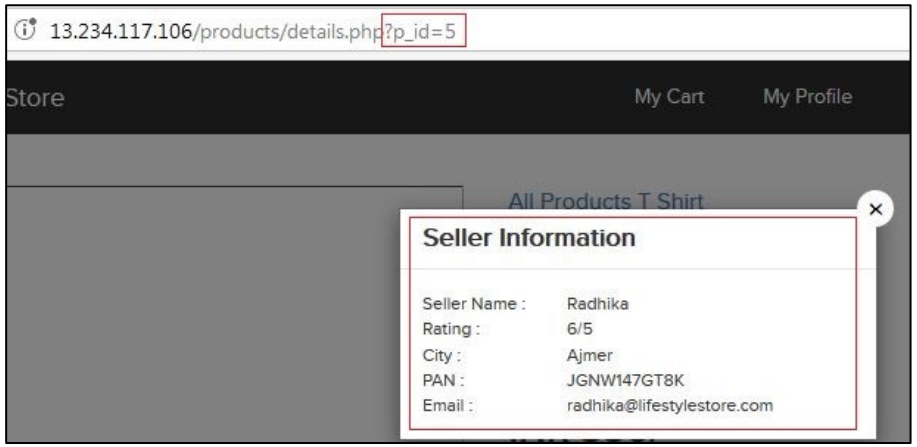
- Log in to your account and navigate to : **http://url/products.php**
- Now select any desired product and click on ‘Seller Info’ option.
- You will be presented with the personal information of the seller which discloses his/her **PAN number**, city and E-mail id.



Observation

- Further more you can access any product by changing the GET based parameter **p_id**





Business Impact – High

- This vulnerability disclose personal information about the sellers including:
  - ❑ PAN Card number
  - ❑ E-Mail ID
  - ❑ City
- The PAN Card number can be used by the hacker to access critical data on sites which make use of such number to register or login.

Recommendations

Take the following precautions to avoid exploitation of PII leakage:

Make sure that the personal information of sellers is well protected and critical data like PAN is not included in the data.

References

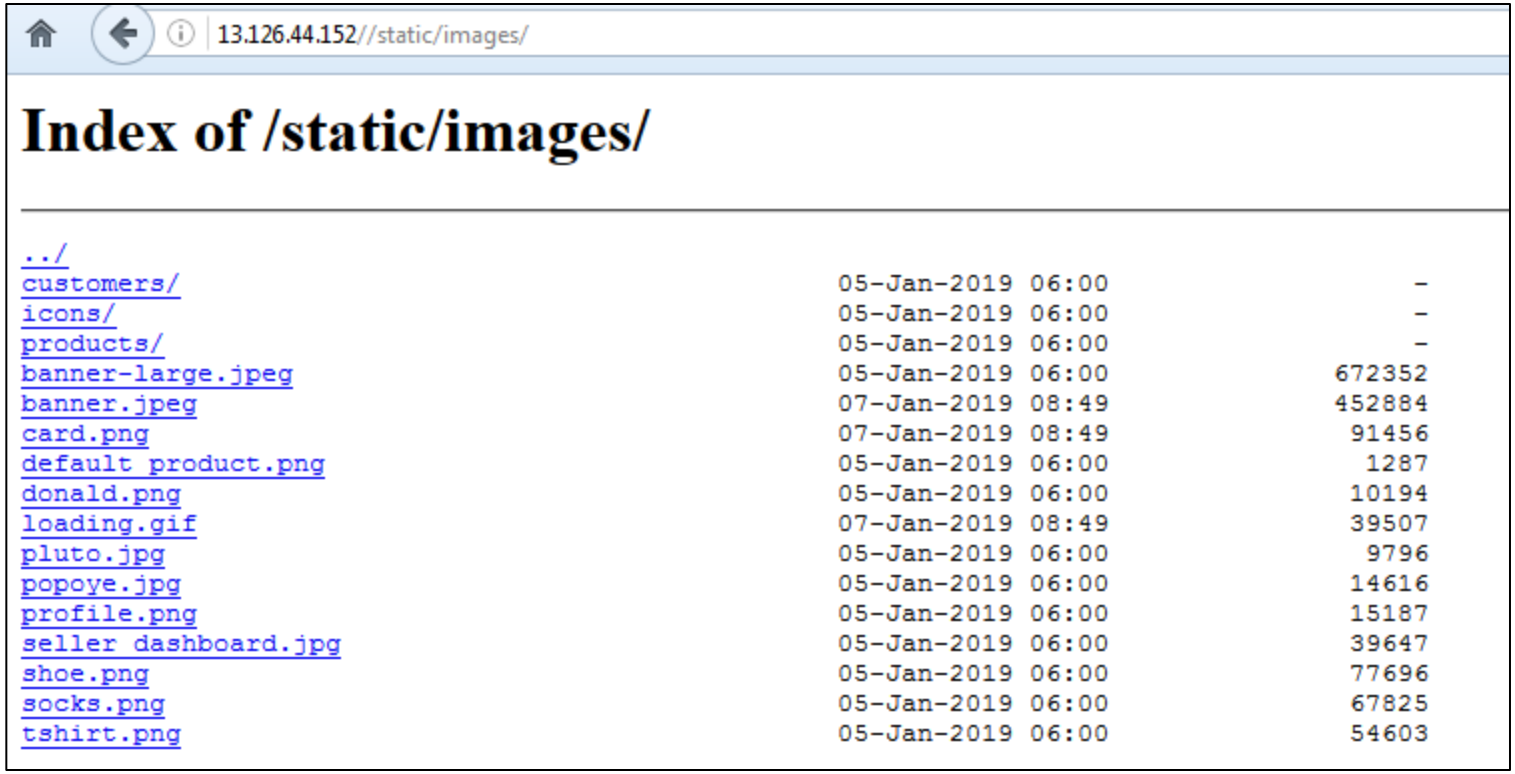
- *https://www.ftc.gov/system/files/documents/public_comments/2016/09/00006-128889.pdf*
- *https://www.cloudcodes.com/blog/what-is-pii.html*
- *https://www.owasp.org/index.php/Forced_browsing*

11. Directory Listing of Configuration Files

Directory Listing of Configuration Files (Moderate)	Below mentioned URL is vulnerable to Directory Listing of Configuration Files. **Affected URL:** ➢ http://url/static/images/ ➢ http://url/ovidentiaCMS/ **Payload:** ➢ http://url/robots.txt

Observation

- Navigate to **http://url/static/images/**
- Complete listing of directory is shown containing names of files and folders of the website.



Observation

- Navigate to **http://url/ovidentiaCMS**
- A complete hidden CMS is accessible which may have some components of known vulnerabilities.

Business Impact – Moderate

- Although this vulnerability does not have a direct impact to users or the server, though it can aid the attacker with information about the server and the users
- Also, attacker can simply download the backups and images and view them.

Recommendations

Take the following precautions:

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

References

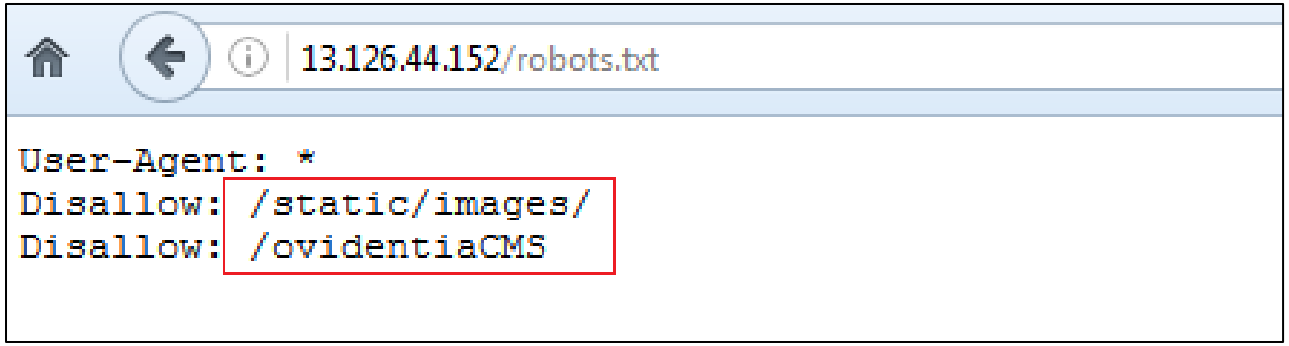
- *https://cwe.mitre.org/data/definitions/548.html*
- *https://www.netsparker.com/blog/web-security/disable-directory-listing-web-servers/*

12. Default Files and Pages

Default Files and Pages (Moderate)	Below mentioned URL is vulnerable to Default files and pages. **Affected URL:** ➢ http://url/robots.txt

Observation

- Navigate to **http://url/robots.txt**
- Which gives us some disallowed URLs.



Business Impact – Moderate

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

Recommendations

Take the following precautions:

- If you have files on your web site that you don't want unauthorized people to access, then configure your server to do authentication, and configure appropriate authorization.

References

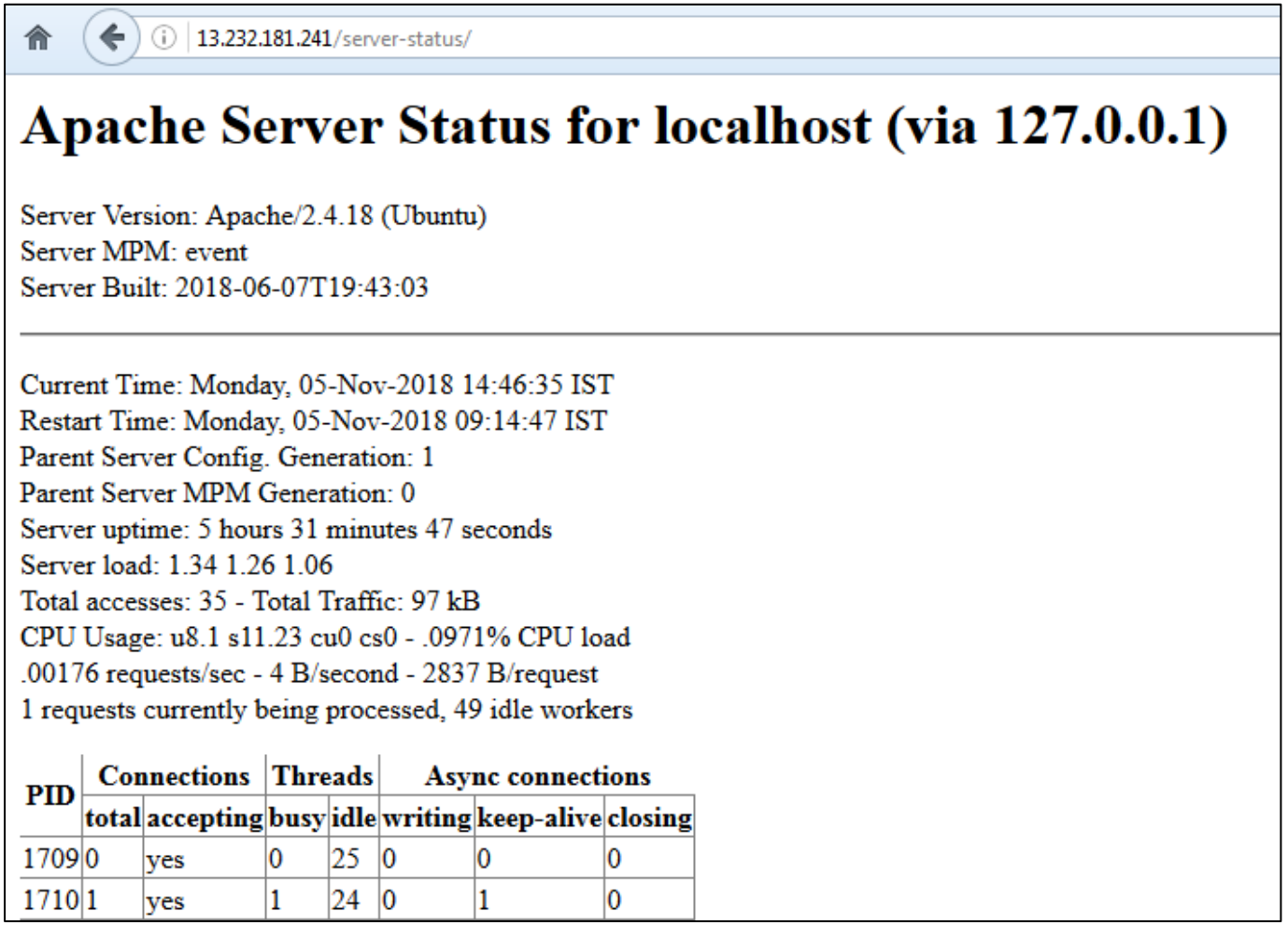
- *http://www.robotstxt.org/faq/nosecurity.html*
- *http://www.robotstxt.org/robotstxt.html*
- *http://www.robotstxt.org/faq/prevent.html*

13. Information Disclosure

Information Disclosure (Low)	Below mentioned URL is vulnerable to Information Disclosure **Affected URL:** ➢ http://url/server-status/

Observation

- Navigate to **http://url/server-status**
- Which gives us the server information on which the website is running on.



Business Impact – Low

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

Recommendations

Take the following precautions:

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

References

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

THANK YOU

For any further clarifications/patch assistance, please contact:

9818210355