Final Term Project

"Detecting Web attacks using Machine Learning"

CS 656

Teammates
Prajwal Mani (pbm6)
Kavya Umashankar (kbu2)
Github:https://github.com/prajwalmani/web_attacks_machine_learning

Table of Contents

Abstract

Internet is fast growing nowadays, as more business activities are being automated and an increasing number of computers are being used to store sensitive information, the need for secure computer systems becomes more apparent. Today's interconnected world makes everyone more susceptible to cyber-attacks. A cyber-attack is a malicious and deliberate attempt by an individual or organization to breach the information system of another individual or organization.

Web attacks are the ones that occur on a website or web applications. Some of the web-based attacks are XSS, Injection attacks, DNS spoofing, Session hijacking, SQLi, Phishing, brute force, DOS and many more. Among these attacks, SQL injection and Crosssite scripting are frequently reported attacks. In these times, the data is highly vulnerable, and the attackers continuously find new ways to illegally retrieve them, as a result, the data should be protected from such attacks. The proposed system detects SQLi and XSS attacks using the Deep Learning approach. A website vulnerable to SQLi and XSS attacks are created using PHP in the front end which is connected to the database. The deep learning algorithms are constructed using python. The query string entered by the user and its corresponding log data is given as input to the Intrusion detection system (Deep Learning algorithm), which detects the given input to be malicious or benign. Based on the result, the user is either allowed to access the website or is blocked.

Introduction

A cyber-attack is any type of offensive action that targets computer information systems, infrastructures, computer networks or personal computer devices, using various methods to steal, alter or destroy data or information systems. It also includes criminal acts like hacktivist protests, harassment and extortion, money laundering, and more.

Web attacks:

It is the attack in which some data will be injected into a web application to manipulate the application and fetch the required information. There are client-side vulnerabilities and server-side vulnerabilities which lead to a web application attack.

There are many web attacks but in this project, we will be working on XSS and SQLi

- **Cross-site scripting (XSS):** This involves an attacker uploading a piece of malicious script code onto a website that can then be used to steal data. Although this strategy is relatively unsophisticated, it remains quite common and can-do significant damage.
- **SQL Injection (SQLI):** This happens when a hacker submits destructive code into an input form. If a system fails to clean this information, it can be submitted into the database where it can change, delete or reveal data to the attacker.

Machine Learning:

Machine learning (ML) is a type of Artificial Intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Motivation

Cybercrime remains a growing challenge in terms of security and privacy practices. Today‘s interconnected world makes everyone more susceptible to cyberattacks. The threat of attacks on web applications to extract data or to distribute malicious code persists. Few security issues of web applications are XSS attacks, SQL injection, Local file inclusion, Path traversal and many more. Some of these attacks are designed to prevent competitors from participating in major events, while others target the complete shutdown of online businesses for months. Effects of these attacks include: Authentication bypass, information disclosure, compromised data integrity, user‘s session hijacking, and access cookies.

Objectives

Cyber Security is an essential component of any company or enterprise across the world, hence the scope of Cyber Security is enormous. Cyber Security is the technology, process, and practice, designed to protect devices, programs, and data from damages, attacks, and other unauthorized access. Many authorized institutions, like the military, government agencies, financial institutions, Banking Sector, etc. have confidential information that is stored on computers and transmitted to networks. With growing cyber-attacks, it has become necessary to protect this sensitive data and personal information.

Here stopping identity theft isn‘t the only goal, but protecting data integrity is also important. As cybercriminals are becoming more advanced, there is a need to understand their change in target, how that is affecting organizations, and their methods used in targeting.

Hardware Requirements

Processor: Intel Core i5
Hard Disk: 50 GB Free Space.
Ram: 8 GB(Minimum)
Display: Monitor(1024*768 pixels) or higher resolution monitor with 32 bit color settings

Software requirements

Operating System: Kali Linux, Windows Programming language:
Back - End: Python Programming language:
Front - End: HTML, PHP, CSS, Bootstrap,JavaScript
Development Environment: Jupyter Notebook/ any other Python IDE
Application Server: XAMPP

Database: MYSQL

Implementation

1) Anaconda Jupyter Notebook
2) VS code

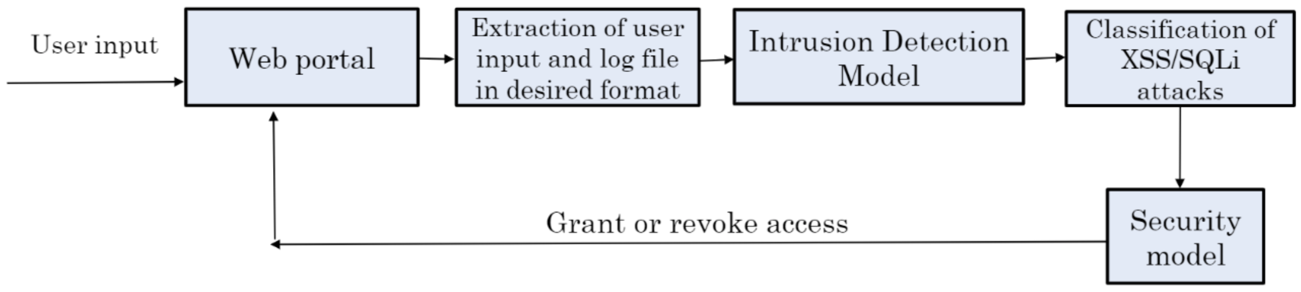
Libraries

1) Numpy
2) Pandas
3) Matplotlib
4) Tensorflow
5) Pickle
6) CV2

Proposed System

A number of approaches have been proposed to detect XSS and SQLi attack vulnerabilities. Some of the existing approaches used machine learning algorithms such as Naïve bayes classifier, ensemble algorithms, decorates algorithms, AD Trees, elastic pooling CNN. There are several drawbacks of these methods and some of them are not compactable with the updated technologies. To overcome these drawbacks, the proposed system uses log based deep learning approach to detect XSS and SQLi attacks. A website vulnerable to SQLi and XSS attacks is created using PHP in the front end which is connected to the SQL database. The deep learning algorithms are constructed using python. Web log file is log file automatically created and maintained by a web server. This contains information about who was visiting the site, where they came from, and exactly what they were doing on the web site.

Model Pipeline


User input
Web portal
Extraction of user input and log file in desired format
Intrusion Detection Model
Classification of XSS/SQLi attacks
Security model
Grant or revoke access

Components

- Web Portal
- User Input extraction and log data
- Intrusion Detection System
- Classification of SQLi/XSS attack
- Security Model

Code Explanation

To view the entire code and its output please use this github link: https://github.com/prajwalmani/web_attacks_machine_learning

The code blocks are not user friendly and the whole code is not mentioned to keep the report the minimal as possible

Here the code below shows how we have constructed a model to detect SQL Injection and XSS attack.

There are two machine learning models used in this project viz MLP and CNN.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer
import keras
```

```python
from keras.models import Sequential
from keras import layers
import tensorflow as tf
from keras.models import model_from_json
from keras.layers import Dense, Activation, Conv2D, MaxPooling2D,Flatten
import cv2
import pickle

def load_csv():
    global sqlidf
    contents=[]
    with open("sqli.csv",'r',encoding = 'utf-8') as f:
        for line in f:
            word=line.split('\n')
            list2 = [x for x in word if x]
            list1 = list2[0].rsplit(',',maxsplit=1)
            sentence=list1[0][1:]
            label=list1[1][:-1]
            listx=[sentence,label]
            contents += [listx]

    contents=contents[1:]
    sqlidf = pd.DataFrame(contents,columns=['Sentence','Label'])

load_csv()


sqlidf['Sentence'] = sqlidf['Sentence'].astype(str)
sqlidf['Label']=sqlidf['Label'].astype(int)

X=sqlidf['Sentence']
y=sqlidf['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

vectorizer = CountVectorizer()
posts = vectorizer.fit_transform(X_train).toarray()
test_posts = vectorizer.transform(X_test).toarray()

input_dim = len(vectorizer.vocabulary_)

model = Sequential()
model.add(layers.Dense(128, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(128,  activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()

classifier_nn = model.fit(posts,y_train,
                    epochs=10,
                    verbose=True,
                    validation_data=(test_posts, y_test))

pickle.dump(vectorizer.vocabulary_, open("dictionary.pickle", 'wb'))

pred=model.predict(test_posts)

for i in range(len(pred)):
    if pred[i]>0.5:
        pred[i]=1
    elif pred[i]<=0.5:
        pred[i]=0

def accuracy_function(tp,tn,fp,fn):

    accuracy = (tp+tn) / (tp+tn+fp+fn)

    return accuracy


def precision_function(tp,fp):

    precision = tp / (tp+fp)

    return precision


def recall_function(tp,fn):

    recall=tp / (tp+fn)

    return recall


def confusion_matrix(truth,predicted):

    true_positive = 0
    true_negative = 0
    false_positive = 0
    false_negative = 0
    for true,pred in zip(truth,predicted):
        if true == 1:
            if pred == true:
                true_positive += 1
            elif pred != true:
                false_negative += 1
         elif true == 0:
            if pred == true:
                true_negative += 1
            elif pred != true:
                false_positive += 1
    accuracy=accuracy_function(true_positive, true_negative, false_positive,
false_negative)
    precision=precision_function(true_positive, false_positive)
    recall=recall_function(true_positive, false_negative)
    confusion_matrix_res = [[true_negative,
false_negative],[false_positive,true_positive]]
return (accuracy,
            precision,
           recall,
           confusion_matrix_res)
accuracy,precision,recall, matrix =confusion_matrix(y_test,pred)
print(" Accuracy : {0} \n Precision : {1} \n Recall : {2} \n Confusion matrix:
{3}".format(accuracy, precision, recall, matrix))
# serialize model to JSON
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save("sqli_model.h5")
print("Saved model to disk")

import sklearn
def testing1(querystring):
    instance = X_test
    instance.iloc[0] = querystring[0]
    vocabulary_to_load = pickle.load(open("dictionary.pickle", 'rb'))
    loaded_vectorizer =
sklearn.feature_extraction.text.CountVectorizer(vocabulary=vocabulary_to_load)
    loaded_vectorizer._validate_vocabulary()
    instance_posts = loaded_vectorizer.transform(instance).toarray()

    pred = model.predict(instance_posts)

    if pred[0]>0.5:
        res=1
    else:
        res=0

    return res

import time
#hello world!
start = time.time()
print(testing1(["105 OR 1=1"]))
stop = time.time()
print("time = ",stop-start)

# load and evaluate a saved model
from numpy import loadtxt
from keras.models import load_model

# load model
model = load_model('model.h5')
# summarize model.
model.summary()

def load_csv():
    contents=[]
    with open("XSS_dataset.csv",'r') as f:
        for line in f:
            word = line.split('\n')
            sentence = word[0]
            index , string = sentence.split(',',maxsplit=1)
            sentence, label = string.rsplit(',',maxsplit=1)
            #sentence = sentence.strip('"')
            contents += [[sentence , label]]


    contents=contents[1:]
    #print(contents)
    global xssdf
    xssdf = pd.DataFrame(contents,columns=['Sentence','Label'])
    xssdf = xssdf.replace({'\t': ''}, regex=True)
    xssdf['Sentence'] = xssdf['Sentence'].astype(str)
    xssdf['Label']=xssdf['Label'].astype(int)


load_csv()
X = xssdf['Sentence']
y = xssdf['Label'].values
trainX, testX, trainY, testY = train_test_split(X,y, test_size=0.2)
train_sentences=trainX.values
test_sentences=testX.values
```

```
def convert_to_ascii(sentence):
    sentence_ascii=[]

    for i in sentence:

        if(ord(i)<8222):      # ”  :  8221

            if(ord(i)==8217): # ’  :  8217
                sentence_ascii.append(134)


            if(ord(i)==8221): # ”  :  8221 ""
                sentence_ascii.append(129)

            if(ord(i)==8220): # “  :  8220
                sentence_ascii.append(130)


            if(ord(i)==8216): # ‘  :  8216
                sentence_ascii.append(131)

            if(ord(i)==8217): # ’  :  8217
                sentence_ascii.append(132)

            if(ord(i)==8211): # –  :  8211
                sentence_ascii.append(133)


            """
            If values less than 128 store them else discard them
            """
            if (ord(i)<=128):
                    sentence_ascii.append(ord(i))

            else:
                    pass


    zer=np.zeros((10000))

    for i in range(len(sentence_ascii)):
        zer[i]=sentence_ascii[i]

    zer.shape=(100, 100)

    return zer
```

```
def preprocessing(sentences):
    arr=np.zeros((len(sentences),100,100))

    for i in range(len(sentences)):

        image=convert_to_ascii(sentences[i])

        x=np.asarray(image,dtype='float')
        image =  cv2.resize(x, dsize=(100,100), interpolation=cv2.INTER_CUBIC)
        image/=128
        arr[i]=image
    return arr

train_arr = preprocessing(train_sentences)
test_arr = preprocessing(test_sentences)
data = train_arr.reshape(train_arr.shape[0], 100, 100, 1)
test_data = test_arr.reshape(test_arr.shape[0], 100, 100, 1)
print("Train data shape : ",data.shape)
print("Test data shape : ",test_data.shape)

model=tf.keras.models.Sequential([

    tf.keras.layers.Conv2D(64,(3,3), activation=tf.nn.relu,
input_shape=(100,100,1)),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128,(3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(256,(3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')

])

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()


num_classes = 2
```

```
train_y = keras.utils.np_utils.to_categorical(data, num_classes)
test_y = keras.utils.np_utils.to_categorical(test_data, num_classes)

batch_size = 128
num_epoch = 10

#model training
model_log = model.fit(data, trainY,
          batch_size=batch_size,
          epochs=num_epoch,
          verbose=1,
          validation_data=( test_data,  testY)
                     )

pred=model.predict(test_data)


for i in range(len(pred)):
    if pred[i]>0.5:
        pred[i]=1
    elif pred[i]<=0.5:
        pred[i]=0

accuracy,precision,recall, matrix =confusion_matrix(testY,pred)
print(" Accuracy : {0} \n Precision : {1} \n Recall : {2} \n Confusion matrix:
{3}".format(accuracy, precision, recall, matrix))

def testing(querystring):
    instance=[]
    instance = testX
    instance = instance[:250]
    instance[-1] = querystring[0]
    test_instance=instance.values
    instance_arr = preprocessing(test_instance)
    instance_data = instance_arr.reshape(instance_arr.shape[0], 100, 100, 1)
    pred=model.predict(instance_data)
    if pred[-1]>0.5:
            res=1
    else:
            res=0

    print(res)


model_json = model.to_json()
with open("xssmodel.json", "w") as json_file:
    json_file.write(model_json)
```

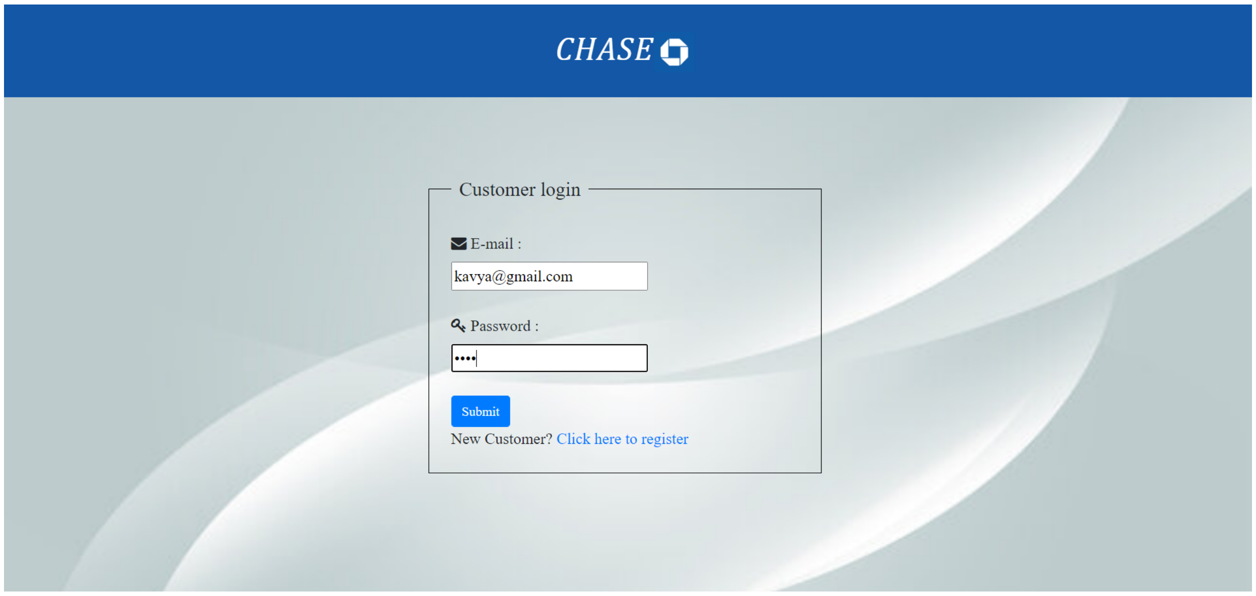
```
# serialize weights to HDF5
model.save("xssmodel.h5")
print("Saved model to disk")
```

Backend to frontend Connection codes:

```
import time
import sys
start=time.time()
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer

import sklearn

from keras.models import model_from_json

import pickle

def load_csv():
    global sqlidf
    contents=[]
    with open("sqli.csv",'r',encoding = 'utf-8') as f:
        for line in f:
            word=line.split('\n')
            list2 = [x for x in word if x]
            list1 = list2[0].rsplit(',',maxsplit=1)
            sentence=list1[0][1:]
            label=list1[1][:-1]
            listx=[sentence,label]
            contents += [listx]

    contents=contents[1:]
    sqlidf = pd.DataFrame(contents,columns=['Sentence','Label'])

load_csv()
# sqlidf = pd.read_csv('sqli.csv')
# In[3]:


sqlidf['Sentence'] = sqlidf['Sentence'].astype(str)
sqlidf['Label']=sqlidf['Label'].astype(int)


# In[4]:


X=sqlidf['Sentence']
y=sqlidf['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# In[5]:
#print(sqlidf)

# load json and create model
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
 # load weights into new model
loaded_model.load_weights("model.h5")
#print("Loaded model from disk")

# In[6]:


def testing1(querystring):
    instance = X_test
    instance.iloc[-1] = querystring[-1]
    # print(instance.iloc[0])
    vocabulary_to_load = pickle.load(open("dictionary.pickle", 'rb'))
    loaded_vectorizer =
sklearn.feature_extraction.text.CountVectorizer(vocabulary=vocabulary_to_load)
    loaded_vectorizer._validate_vocabulary()
    instance_posts = loaded_vectorizer.transform(instance).toarray()

    pred = loaded_model.predict(instance_posts)
    # print(pred[0])
    if pred[-1]>0.5:
        res=1
    else:
        res=0
    return res
```

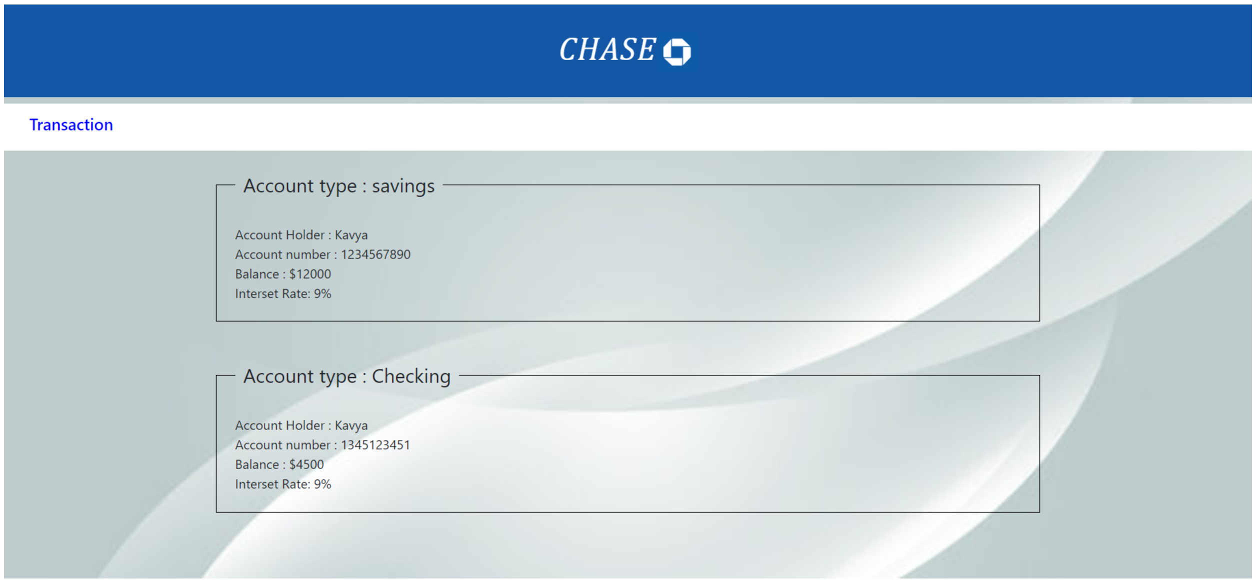
Observations

Output with Correct information given



kavya@gmail.com

••••



CHASE

Transaction

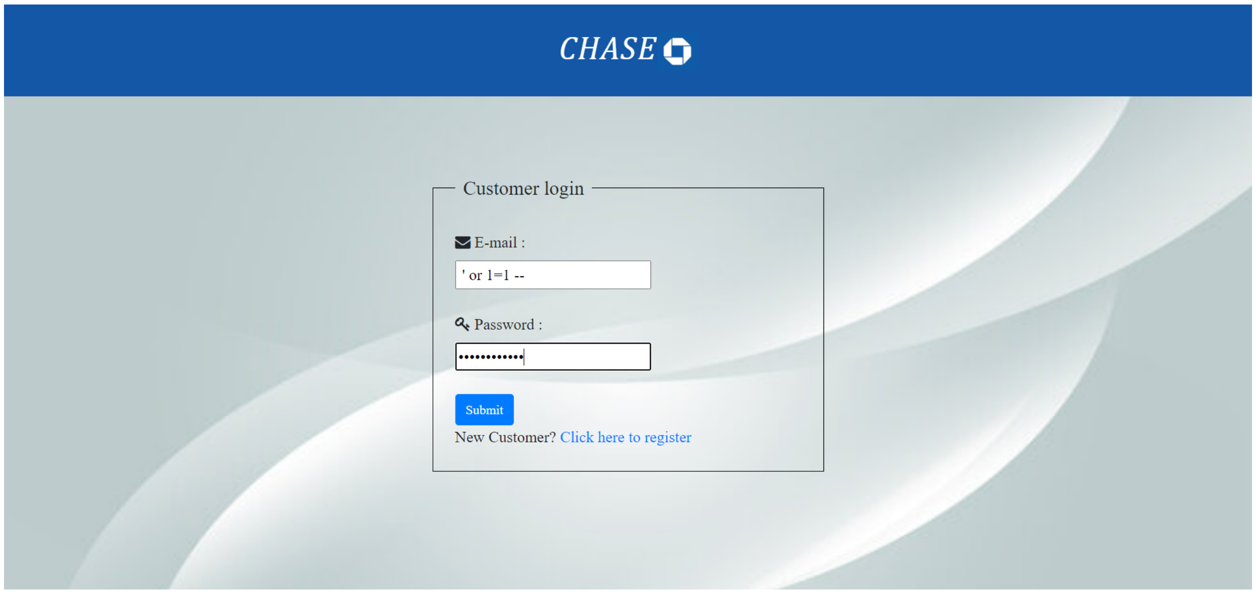
Account type : savings

Account Holder : Kavya
Account number : 1234567890
Balance : $12000
Interset Rate: 9%

Account type : Checking

Account Holder : Kavya
Account number : 1345123451
Balance : $4500
Interset Rate: 9%

With login SQLI Bypass



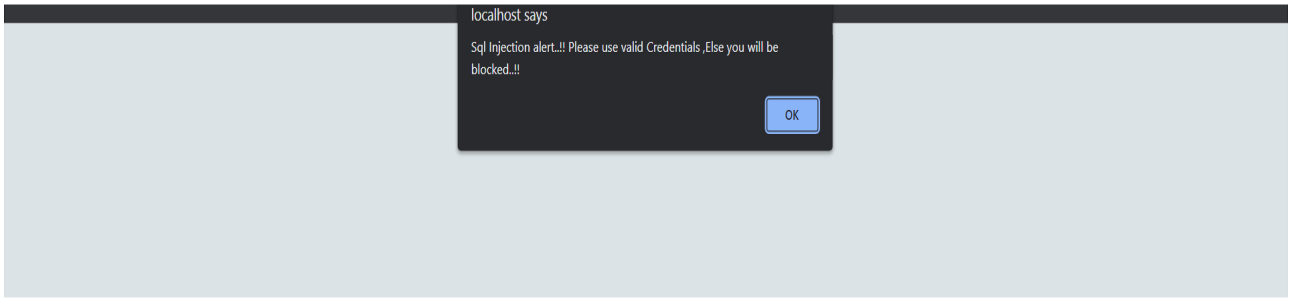
CHASE

Customer login

E-mail : ' or 1=1 --

Password : ••••••••••••

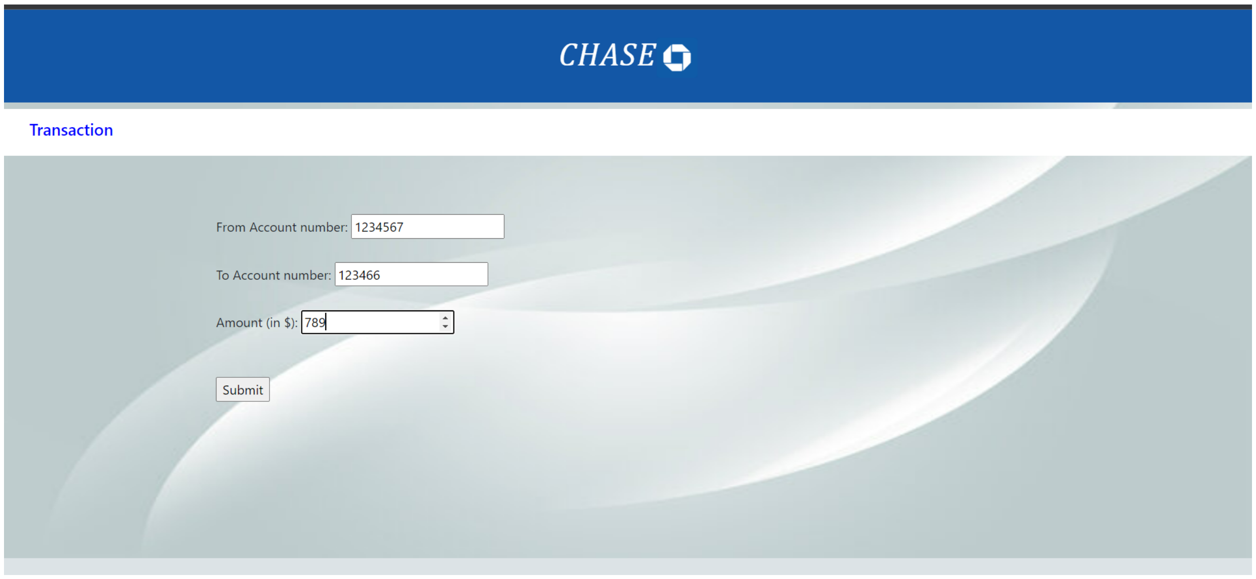
New Customer? Click here to register



localhost says

Sql Injection alert..!! Please use valid Credentials ,Else you will be blocked..!!

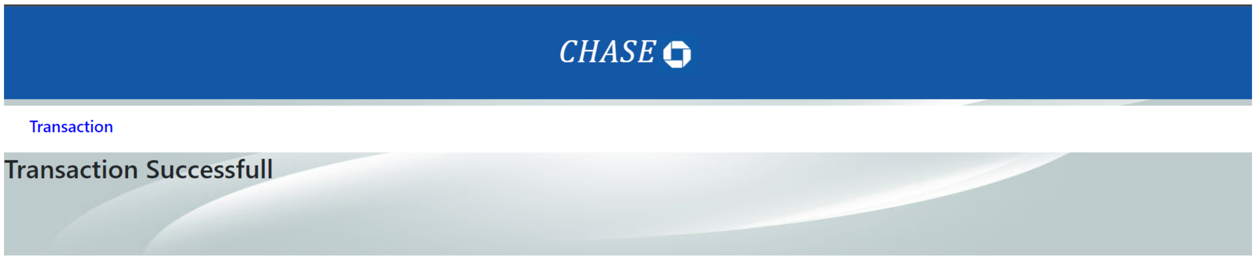
Without XSS Attack:



From Account number: 1234567

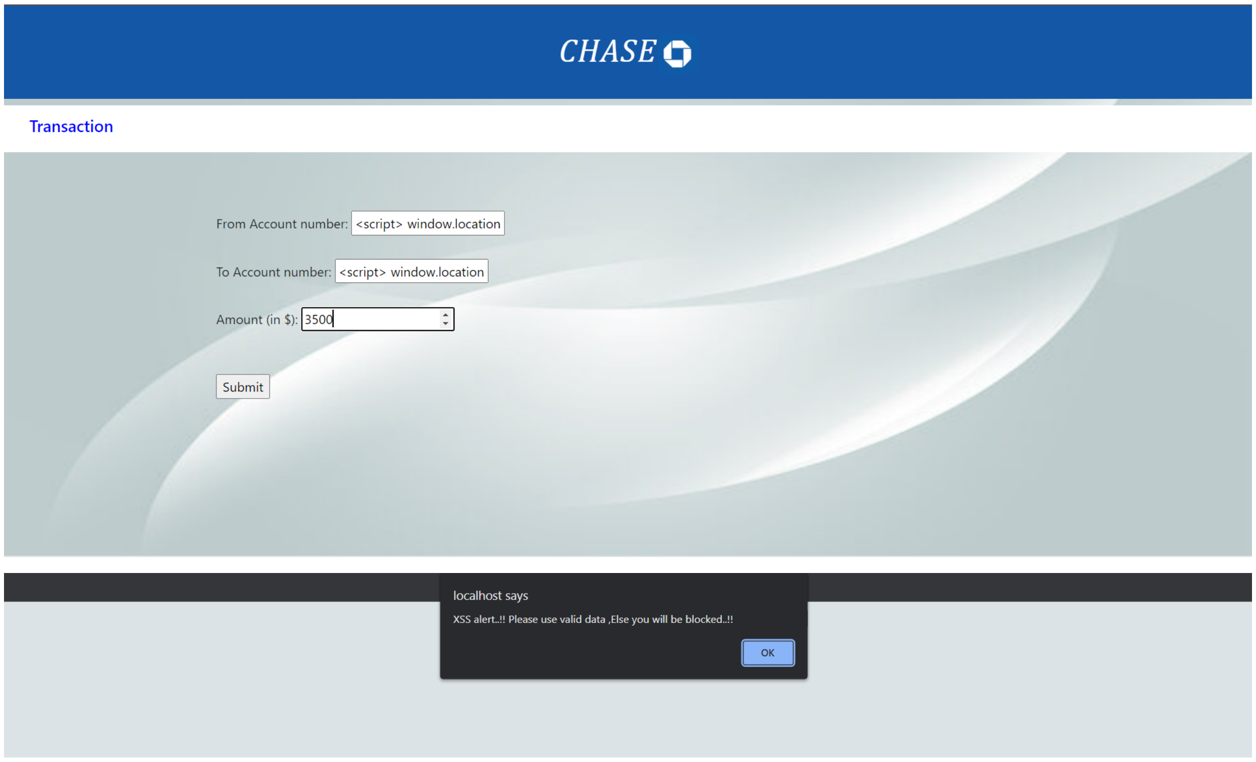
To Account number: 123466

Amount (in $): 789



Transaction Successfull

XSS attack:



From Account number: <script> window.location

To Account number: <script> window.location

Amount (in $): 3500

localhost says

XSS alert..!! Please use valid data ,Else you will be blocked..!!

Conclusion:

Cyber-crime is constantly on the rise, and many smaller businesses are extremely vulnerable as a result of ineffective cyber security due to enormous usage of internet. The impact of cyber-crime can range from financial losses to business losses, due to which different industries can also suffer many disastrous consequences as a result of criminal cyber-attacks. SQL injection can leave the application at a high-risk of compromise resulting in a threat to the integrity, and confidentiality of data as well as authorization and authentication aspects of the application.

A successful attack may end in the unauthorized viewing of user lists, the deletion of tables and, in certain cases, the attacker gaining admin rights to a database, all of which are highly deleterious to a business. The effect of XSS attack ranges from user's Session Hijacking to disclosure of sensitive data, Cross site request forgery (CSRF) attacks and other security vulnerabilities. In times like these it is crucial to protect the sensitive personal and business information through prevention, detection and response to different online attacks. To overcome the disadvantages of traditional machine learning approach, the proposed method uses web log files and deep learning algorithm to detect SQLi, XSS attacks.

The proposed system uses deep learning algorithm which scans the data to search for features that correlate and combine them to enable faster learning without being explicitly told to do so. The server generates a log file automatically created and maintained by a web server. Every time any browser or user-agent, Google included, requests any resource—pages, images, java script file from the server, the server adds a line in the log. In relation to cyber security, log data points out to the red flags in the systems such as unusual behaviour, unauthorized access, extreme traffic and any suspicious activities by the attackers.