Experiment No.: 01

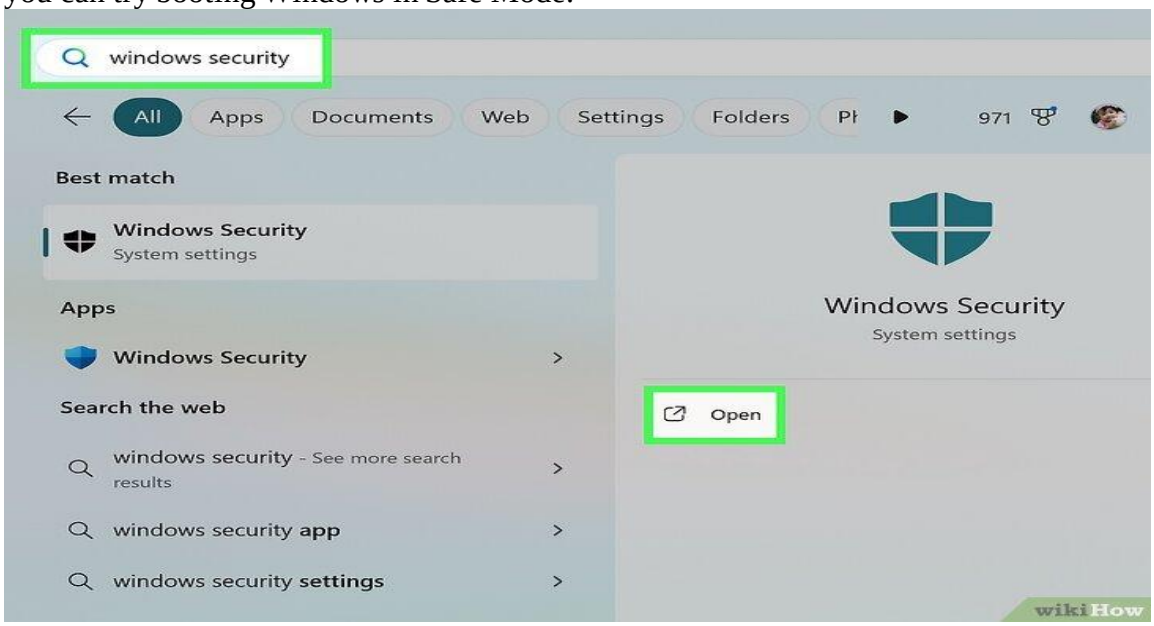
Title: Identify malwares and viruses from your system by using any malware/virus detection tool.

Steps:

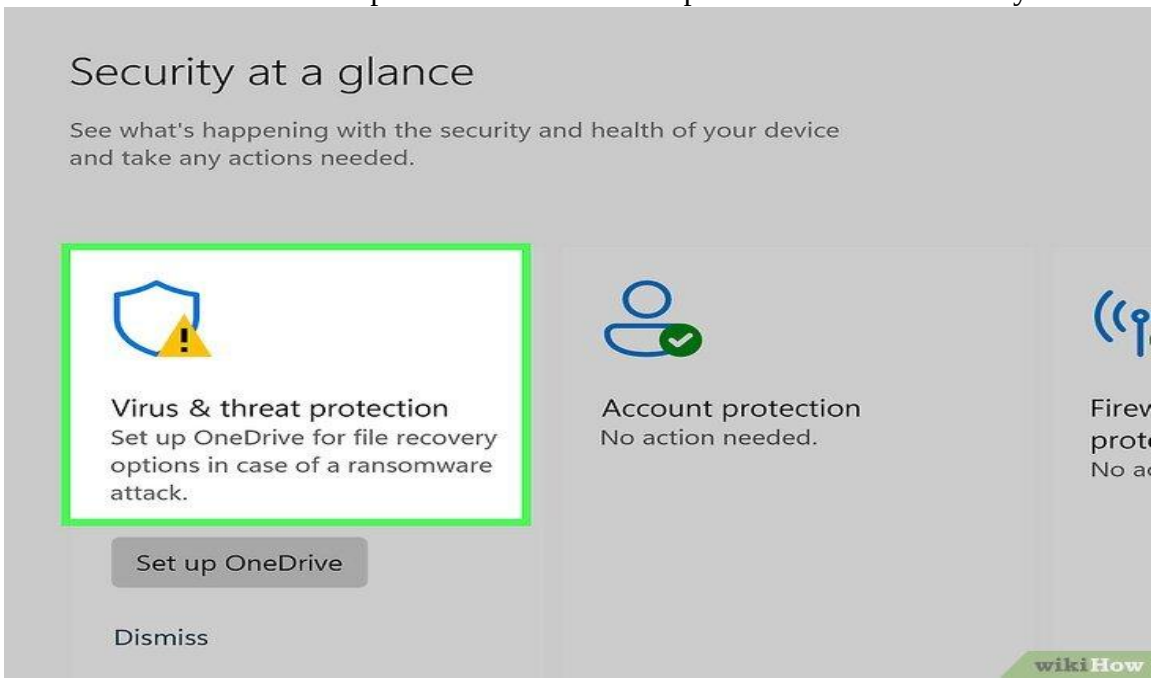
1. Open Windows Security.

Windows Security comes pre-installed on all versions of Windows 10 and 11. It includes virus protection, which you can use to scan for viruses and malware. Enter "Windows Security" in the Windows search bar and then click Windows Security.

If you are having a hard time booting into Windows, or Windows is running prohibitively slow, you can try booting Windows in Safe Mode.



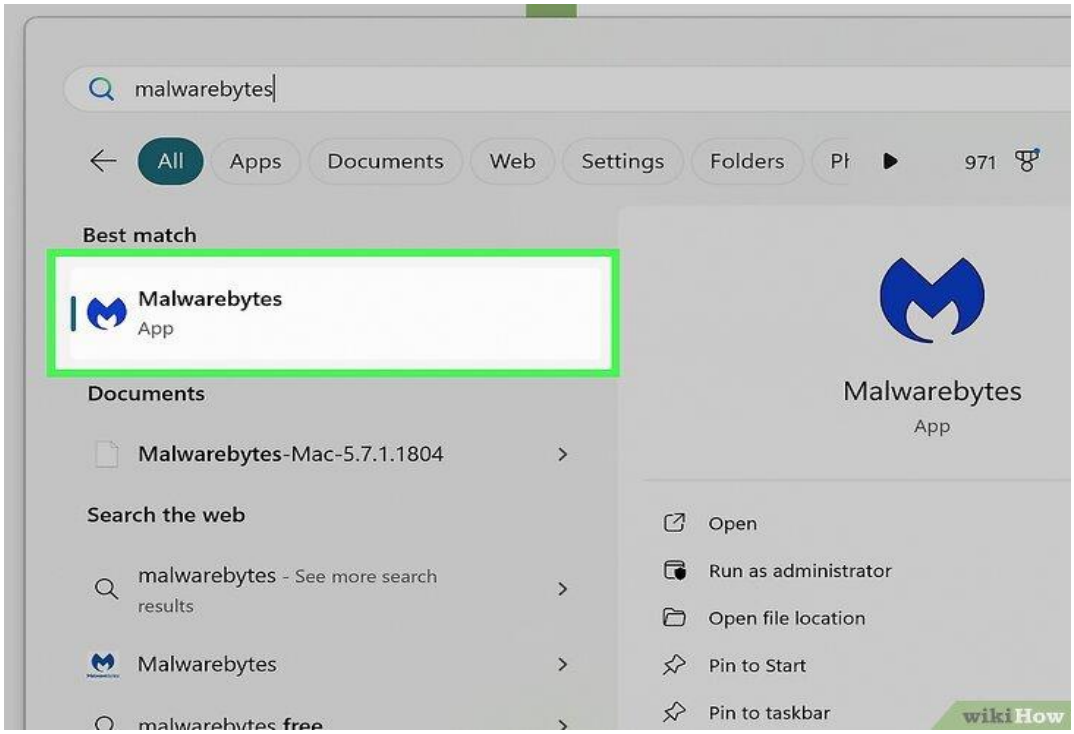
2. Click Virus & threat protection. It's the first option in Windows Security.



3. Click Quick Scan or Scan Now. This will begin scanning your computer for viruses and malware. If any threats are detected, follow the prompts to remove or quarantine the threats.

4. Download and install Malwarebytes. It is a reputable antivirus/ antimalware that is available for Windows and macOS. You can use the free version of Malwarebytes to scan for viruses on your computer. Use the following steps to download and install Malwarebytes:
   Go to https://www.malwarebytes.com/mwb-download in a web browser.
   Click Free Download.
   Open the installation file in your Downloads folder.
   Follow the prompts.

5. Open Malwarebytes. You can open Malwarebytes in the Windows Start menu or the Applications folder on Mac.



6. Click Scan. This will begin scanning your computer for viruses and malware. When the scan is complete, follow the prompts to remove or quarantine any threats detected.

Conclusion:

I have successfully learned how to identify malwares and viruses from my system by using Malwarebytes detection tool.

Experiment No.: 02

Title: Use keylogger to get confidential data.

Theory:

Keyloggers:

A keylogger (keystroke logger) is a tool that records the keys a user types on a keyboard. It can capture passwords, messages, credit-card numbers and any other typed information.

Types:

1. Software keyloggers: Programs that run on the device (can be installed as malware or as legitimate monitoring software).
2. Hardware keyloggers: Physical devices placed between keyboard and computer or built into the keyboard.
3. Firmware/firmware-level keyloggers: Hidden in device firmware or low-level system software (harder to detect).

How they work (high level):

Keyloggers intercept keystrokes by hooking into the operating system's input handling, logging characters to a file, or sending them to a remote server. Hardware variants simply record electrical signals from the keyboard.

Uses:

Malicious: Stealing credentials, identity theft, corporate espionage.
Legitimate (controlled): Parental controls, employee monitoring (with consent and legal compliance), forensic investigations.

Signs of infection / indicators:

Unexplained slowdowns or crashes.
Unknown processes/services running.
Unexpected outbound network traffic (especially to unknown hosts).
New files or scheduled tasks that you didn't create.
Physical tampering of keyboard/cables.

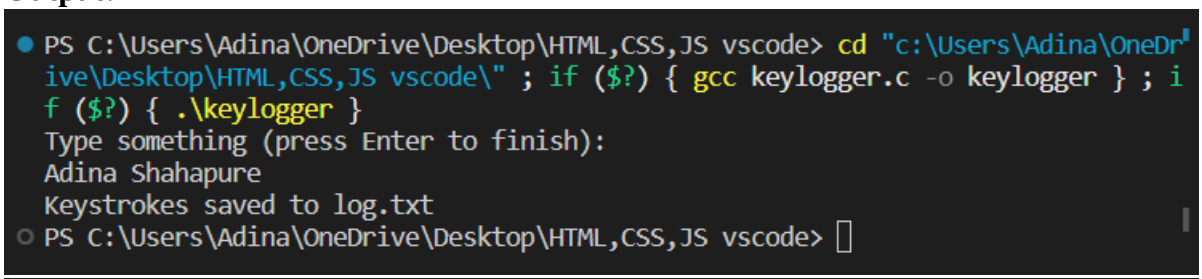
Detection & prevention (safe, defensive steps):

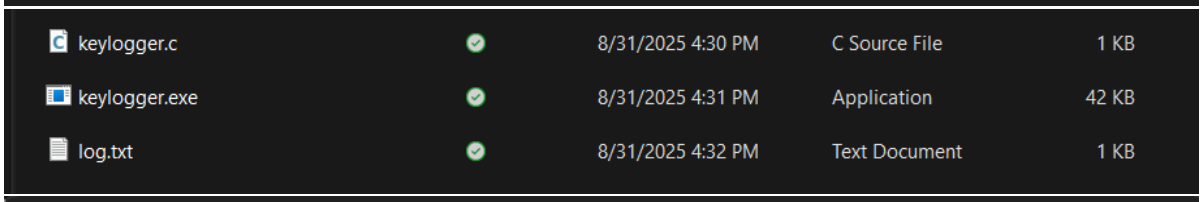
Install reputable antivirus/anti-malware and keep definitions up to date.
Use a software firewall and monitor outbound connections.
Keep OS and applications patched.
Avoid downloading/installing software from untrusted sources.
Use multi-factor authentication (MFA) so stolen passwords alone are not enough.
Use a password manager (it autofills and reduces typing of credentials).
Limit physical access to devices and inspect keyboard connections occasionally.
Employ least-privilege accounts (don't use admin account for daily tasks).
For high-security needs, consider endpoint detection & response (EDR) tools and regular security audits.

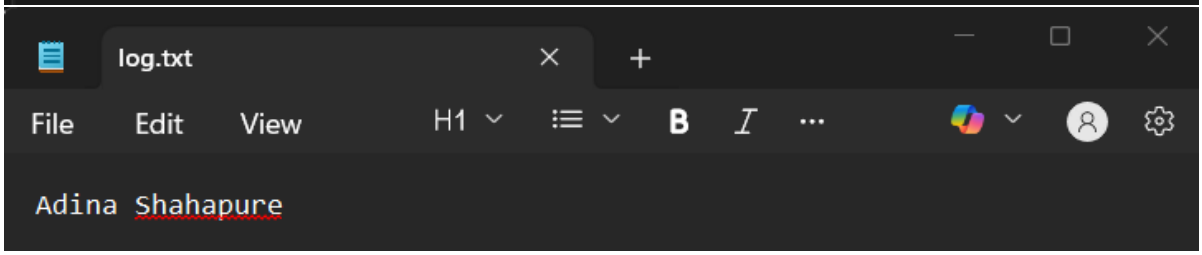
Code:

```
#include <stdio.h>
int main() {
   char a;
   FILE *f = fopen("log.txt", "w");
   if (f == NULL) {
      printf("Error opening file!\n");
      return 1;
      }
   printf("Type something (press Enter to finish):\n");
   while ((a = getchar()) != '\n') {
      fputc(a, f); } // log character
   fclose(f);
   printf("Keystrokes saved to log.txt\n");
   return 0;
   }
```

Output:







Conclusion:

I have successfully created a safe, simulated keylogger in C that captures and stores user keystrokes in a file.

Experiment No.:04

Title: Implement Caesar cipher algorithm.

Theory:

The Caesar cipher is a type of substitution cipher where each letter in the original message (the plaintext) is replaced by a letter a fixed number of positions down the alphabet. This fixed number is called the key.

For example, with a key of 3:

Each 'A' becomes 'D'

Each 'B' becomes 'E'

Each 'C' becomes 'F'

... and so on.

After 'Z', the alphabet wraps around to the beginning, so:

'X' (23) + 3 = 26, which becomes 'A' (0)

'Y' becomes 'B'

'Z' becomes 'C'

Simple Example:

Let's encrypt the word "HELLO" with a key of 3.

H (7) + 3 = 10 → K

E (4) + 3 = 7 → H

L (11) + 3 = 14 → O

L (11) + 3 = 14 → O

O (14) + 3 = 17 → R

Result: The encrypted word (ciphertext) is "KHOOR".

To decrypt "KHOOR" back to "HELLO", you would shift each letter up the alphabet by the same key (3).

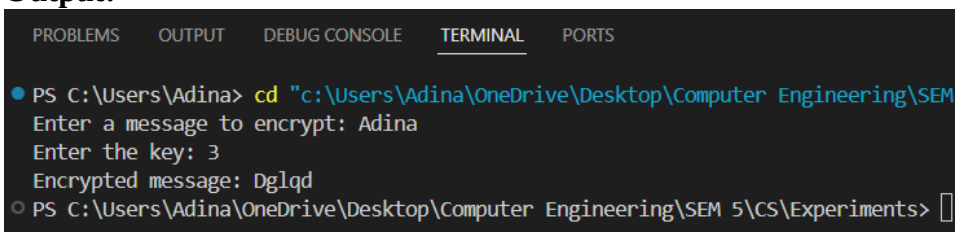
Algorithm:

1. Start: Begin the program.
2. Get Input:
    - Ask the user to type a message and store it.
    - Ask the user to type a secret number (the key) and store it.
3. Process Each Letter: For every single character in the message, do the following:
    a. Look at the current character.
    b. If it's an uppercase letter (A-Z):
        - Find its position in the alphabet (A=0, B=1, ..., Z=25).
        - Add the key to this position.
        - If the new position is 26 or more, wrap around to the start of the alphabet (e.g., position 26 becomes 0, 27 becomes 1). The % 26 (modulus) operation does this.
        - Convert this new number back to an uppercase letter.
    c. Else, if it's a lowercase letter (a-z):
        - Do the same process as above, but for lowercase letters.
    d. Else, if it's anything else (like a space, number, or symbol):
        - Print an error message and stop the program immediately.
4. Replace the Letter: Put the new, encrypted letter back into the message, replacing the old one.
5. Finish: After all letters have been processed, print out the fully encrypted message.
6. End: Stop the program.

Code:

```
#include <stdio.h>
int main()
{
   char text[500], ch;
   int key;
   printf("Enter a message to encrypt: ");
   scanf("%s", text);
   printf("Enter the key: ");
   scanf("%d", &key);
   for (int i = 0; text[i] != '\0'; ++i)
   {
      ch = text[i];
      if (ch >= 'A' && ch <= 'Z')
      {
         ch = (ch - 'A' + key) % 26 + 'A';
      }
      else if (ch >= 'a' && ch <= 'z')
      {
         ch = (ch - 'a' + key) % 26 + 'a';
      }
      else
      {
         printf("Invalide Message: \n");
         return 1;
      }
      text[i] = ch;
   }
   printf("Encrypted message: %s\n", text);
   return 0;
}
```

Output:



Conclusion: I have successfully implemented Caesar cipher algorithm using C program.