Keylogger Project Report

For this project, our group wanted to get a hands-on look at how attackers create simple tools that collect sensitive information from a machine without the user noticing. We have learned about malware and keyloggers in class, but actually building one ourselves gives a much better understanding of how they work in practice. We chose to make a keylogger because it is a common and simple example of an attack tool that silently records whatever a user types. Our goal was not to build anything very complex or over the top. Instead, we wanted to break down the basics and understand how each part functions. Since we ran everything inside our isolated Proxmox environment, we were able to test and experiment safely without putting any real systems at risk.

The main purpose of the project was to build a tool that could capture keystrokes, store them in a readable log, and then exfiltrate that log to an external machine we controlled. Seeing how these pieces worked together showed us how little code an attacker actually needs to collect private information. We also wanted to understand how attackers structure their tools and what kinds of decisions go into something as simple as a keylogger. Working through it step by step gave us a practical look at the offensive side of cybersecurity and helped us gain a deeper understanding of how these tools operate. By keeping the project focused on the fundamentals, we were able to clearly see each stage of the attack flow. This included how keystrokes are captured at the system level, how the logs are stored, how special keys are handled, how the program reacts to triggers like the ESC key, and how the stolen data gets sent off the machine through a network channel. Putting all of these steps together gave us a

clearer understanding of how attackers gather sensitive information in a simple but effective way, which was the main goal of the project.

The technical part of our project started with setting up two virtual machines inside our Proxmox environment. We ran the keylogger on Kali Linux because it gave us a safe place to run Python scripts and interact with system input. Our Ubuntu VM acted as the attacker machine where the logs would be sent. To prepare the machines, we installed the pynput library on Kali so the script could capture keyboard presses, and we installed the vsftpd FTP server on Ubuntu so it could receive the log file over the network. Once the environment was ready, we built the keylogger in Python. The main part of the script used the “pynput.keyboard.Listener” class, which watches the keyboard and triggers functions when keys are pressed or released. Every time a key was pressed, our “on_press” function added the key to a list, increased a counter, and printed the key in the terminal. This made it easier to confirm that everything was working while we developed the script. By tracking a counter, we were able to set the script to write each key to the log file as soon as it was pressed so that the log file updated in real time.

The hardest part of capturing keystrokes was formatting the log file so it was clean and readable. Printable characters show up in pynput as strings with single quotes, and special keys like enter, shift, backspace, and tab show up as “Key.something”. We wrote logic to handle each of these differently. Printable characters had their quotes removed before being written to the file. Spaces were turned into newline characters to make the output easier to read. Other special keys were wrapped in brackets so they appeared as “[esc]” or “[alt]”. We ignored shift so that the log was

not filled with unnecessary entries. If there are capital letters in the log file, then we know the user pressed shift. Exfiltration happened in a function that we set up to be easily executable for demonstration purposes. When the user pressed ESC, the “on_release” function triggered the exfiltration process. It opened an FTP connection to the Ubuntu VM using hardcoded credentials and uploaded the keylogs.txt file with Python’s ftplib library. After the file was uploaded, the script ended by returning False from the “on_release” function, which shut down the listener and stopped the program. The full attack includes collecting keystrokes, saving them, sending them to a remote machine, and then shutting itself down.

After we finished the keylogger, we ran several tests to make sure everything worked. We typed normal sentences, numbers, special characters, and punctuation like periods to confirm that the script captured and formatted everything correctly. We also checked that pressing ESC always uploaded the log file to Ubuntu and that the uploaded version matched the original file. Throughout all our tests, the script worked consistently, and the attack worked exactly the way we planned.

We chose this project because keyloggers are one of the simplest and most effective tools attackers use. Even though there are more advanced types of malware, a lot of real attacks still rely on basic ideas like capturing user input and quietly stealing it. Building our own version gave us a better understanding of how something this simple can still have a big impact. It also showed us that attackers do not always need complicated techniques. Sometimes a basic script that runs in the background and sends keystrokes to another machine is enough to cause a lot of damage.

We used Python because it gave us everything we needed without making the project more complicated than it had to be. The pynput library let us capture keystrokes without dealing with low-level input hooks, and ftplib gave us a simple way to send the log file over FTP. Our goal was to focus on the attack process, not the low-level programming details. Python was also easier for us to read, test, and explain. We chose FTP for exfiltration because it clearly showed the idea of stolen data leaving the machine. While real attackers will use encrypted or hidden channels, FTP worked well inside our Proxmox environment and made it easy to separate the attacker machine from the compromised one since the Ubuntu VM acted as the receiving server.

Breaking the attack into smaller parts helped us understand why each step mattered. Seeing how keystroke capture works at the system level showed us how input hooks can be misused if a machine is not secured. Watching the log file appear on the Ubuntu server made it clear how fast data can leave a system with almost no user interaction. Even tying the cleanup behavior to the ESC key showed us how attackers want their tools to exit cleanly so they do not draw attention. The project also gave us important defensive insight. Since our script used very few resources and produced almost no visible output, it showed us how easily a simple tool like this can go unnoticed if there is no proper monitoring. Building it ourselves gave us a more realistic idea of what defenders deal with in real environments.

In conclusion, our group was able to design, build, and test a working keylogger inside a lab environment. The project let us walk through each stage of a real attack, starting with capturing keystrokes and ending with sending the stolen data to a remote machine. While doing this, we learned how keystroke listeners work, how different keys

need to be formatted so the log is readable, how logs are stored and exfiltrated, and how attackers structure their tools so they run quietly and behave in a predictable way. The project showed us that it does not take much Python code to create a fully functional keylogger. Seeing the logs show up on our Ubuntu VM made the whole process feel real and helped us understand how impactful a simple tool like this can be in a real environment. It also made us more aware of why system hardening, network monitoring, and least privilege are necessary when defending against these types of attacks. The project was successful overall and we are happy with the results. More importantly, we ended up with a clearer understanding of how simple attacker tools can be and why it is important to detect the signs of data collection and exfiltration. This hands-on experience will help us in real environments where understanding attacker techniques matters.