

# *SniffUSB: USB Surveillance and Alerting Tool for Unauthorized Device Detection*

"A real-time USB monitoring and alert system with screenshot-based evidence and email-based permission control."

Name: Uday HN

College: Rashtriya Raksha University, Gujarat

Course: MSc in Cyber Security and Digital Forensics

LinkedIn: [linkedin.com/in/udayhoratti](https://www.linkedin.com/in/udayhoratti)

GitHub: [github.com/udayhoratti](https://github.com/udayhoratti)

## *MY Project Description*

In many secure environments, unauthorized USB device usage can lead to data theft or malware injection. This project aims to detect USB device insertions in real-time and send an **email alert to the admin** requesting permission to allow or deny access. The system runs on **Kali Linux** and uses **Python, udev rules, and Gmail SMTP**.

## *Objective*

- Detect any USB device connected to the system.
- Automatically extract basic device information.
- Send an **email alert** to the admin with permission request.
- Create a **low-cost**, real-time alert system without using external hardware.

## *Tools & Technologies Used*

This project was developed on **Kali Linux**, a security-focused Linux distribution used as the base operating system. The alert logic was implemented using **Python 3**, a powerful and user-friendly scripting language. To detect USB insertions in real-time, **Udev Rules** were configured, which monitor hardware changes on the system. For sending email alerts when a USB device is connected, the script uses **smtplib**, a built-in Python library that interfaces with Gmail's SMTP server. To ensure the alert script runs automatically at boot or at scheduled intervals, **crontab** or **systemd** was used for automation.

## *Purpose:*

To detect unauthorized USB device connections on a Linux machine (like Kali) and instantly

send an email alert with details, so that a system admin can allow or deny access. This helps protect from USB-based malware or data theft.

So, my main intension is to:

Whenever a USB device is inserted into your Kali Linux system, the script captures its details and immediately sends you an email asking for permission.

Now I'm installing required tool

Mailutils

**mailutils** is a package in Linux used for sending emails directly from the terminal or shell scripts. It provides the mail command, which allows you to send **simple text-based emails** to local or external email addresses.

```
Installing:
mailutils

Installing dependencies:
gsasl-common guile-3.0-libs libgsasl18 libgssglue1 libmailutils9t64 libntlm0 mailutils-common

Suggested packages:
mailutils-mh mailutils-doc

Summary:
Upgrading: 0, Installing: 8, Removing: 0, Not Upgrading: 460
Download size: 9388 kB
Space needed: 63.6 MB / 2626 MB available

Continue? [Y/n] y
Get:1 http://kali.download/kali kali-rolling/main amd64 gsasl-common all 2.2.2-1.1 [52.7 kB]
Get:2 http://http.kali.org/kali kali-rolling/main amd64 guile-3.0-libs amd64 3.0.10+really3.0.10-4 [6877 kB]
Get:3 http://kali.download/kali kali-rolling/main amd64 libgssglue1 amd64 0.9-1.1 [20.5 kB]
Get:4 http://kali.download/kali kali-rolling/main amd64 libntlm0 amd64 1.8-4 [22.4 kB]
Get:5 http://kali.download/kali kali-rolling/main amd64 libgsasl18 amd64 2.2.2-1.1 [80.1 kB]
Get:6 http://kali.download/kali kali-rolling/main amd64 mailutils-common all 1:3.19-1 [818 kB]
Get:7 http://kali.download/kali kali-rolling/main amd64 libmailutils9t64 amd64 1:3.19-1 [943 kB]
Get:8 http://kali.download/kali kali-rolling/main amd64 mailutils amd64 1:3.19-1 [574 kB]
Fetched 9388 kB in 5s (1709 kB/s)
```

- To **send email alerts** or notifications directly from the terminal or scripts.
- It can be used as an alternative to smtplib in Python when you want to send emails using shell commands.
- It works well with tools like cron or systemd to notify users of system events.

## ← App passwords

App passwords help you sign into your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.

[Learn more](#)

You don't have any app passwords.

To create a new app specific password, type a name for it below...

App name  
USB\_ALERT

Create

```
GNU nano 8.4 /home/uday/usb_alert.py *
import smtplib
import sys
from email.mime.text import MIMEText

device_info = sys.argv[1]

msg = MIMEText(f"A USB device was inserted:\n\n{device_info}\n\nAllow it? [Yes/No]")
msg['Subject'] = 'USB Alert - Permission Required'
msg['From'] = '24mcsdf056@student.rru.ac.in'
msg['To'] = '24mcsdf056@student.rru.ac.in'

server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login('24mcsdf056@student.rru.ac.in', '24mcsdf056@student.rru.ac.in')
server.send_message(msg)
server.quit()
```

This is my python script where it will send me a mail and also in that I have added my mail id and password

In this main working part involves that

Setup Udev Rule (USB Detection Trigger) this will detect USB insertions and trigger my script

So, in order to automate this process, I will add another small script in order to make it more realistic

```
GNU nano 8.4 /etc/udev/rules.d/MY_USB.rules
ACTION="add", SUBSYSTEM="usb", ATTR{idVendor}="*", ATTR{idProduct}="*", RUN+="/usr/bin/python3 /home/uday/usb_email_trigger.py"
```

/etc/udev/rules.d/MY\_USB.rules in this path

This is the core part it will run your usb\_email\_trigger.py file whenever any USB is inserted.

Now for the time being Creating Trigger Script

```
GNU nano 8.4 /home/uday/usb_email_trigger.py
import os
import subprocess

usb_info = subprocess.getoutput("lsusb | tail -n 1")
subprocess.call(["python3", "/home/uday/usb_alert.py", usb_info])
```

This is the normal pattern where I will get an usb details and also making it fully executable with high privilege 777 executable

Its time for testing lets Reload Udev Rules

## LET'S TEST

```
(root@uday)-[/home/uday]
# lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	0	30G	0	disk	
├─sda1	8:1	0	29G	0	part	/
├─sda2	8:2	0	1K	0	part	
└─sda5	8:5	0	975M	0	part	[SWAP]
sr0	11:0	1	4G	0	rom	

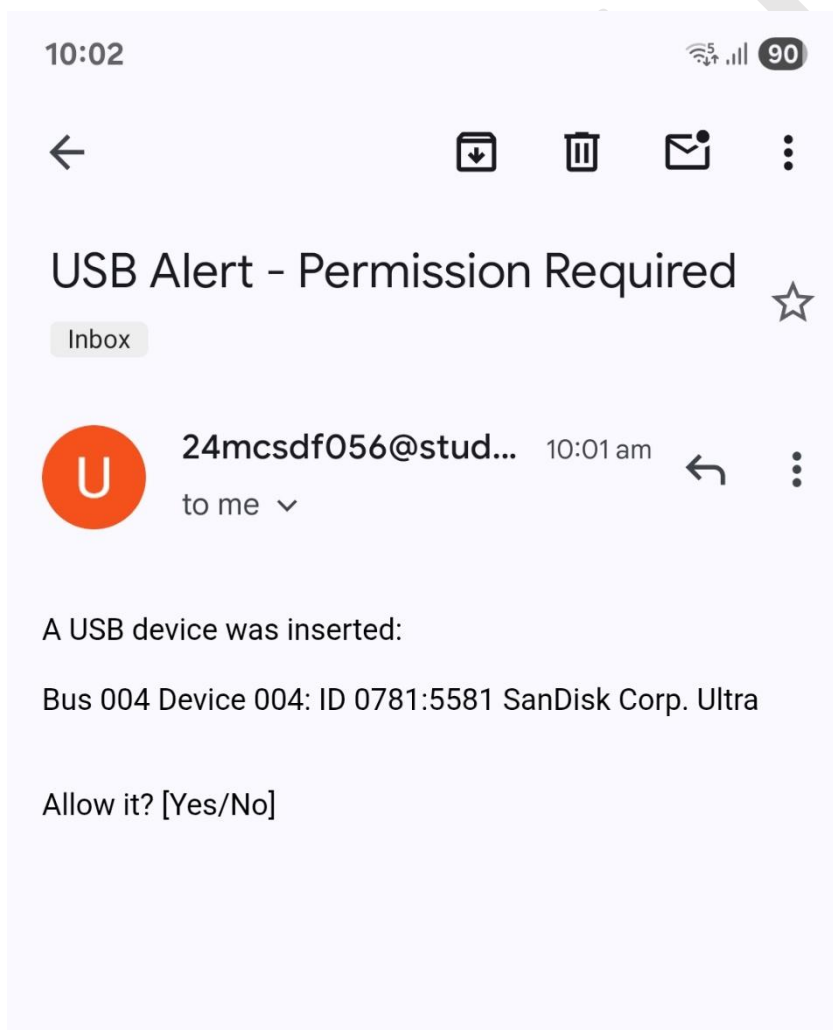
This is before plugin ufs

```
(root@uday) - [ /home/uday ]
# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda          8:0    0   30G  0 disk
├─sda1       8:1    0   29G  0 part /
├─sda2       8:2    0    1K  0 part
└─sda5       8:5    0   975M  0 part [SWAP]
sdc          8:32    1 28.7G  0 disk
└─sdc1       8:33    1 28.7G  0 part
sr0         11:0    1    4G   0 rom
```

I Plugged on USB device and it is detected

I just Wait 5–10 seconds.

There it goes I received an email like finally it works





## Sample Output

Ok for the same lets enhance little bit advance I will try to get an screen shot of the same on the desktop at the time of usb insertion

Again, I need one tool called **scrot** to take screenshot

```
(root@uday)~[/home/uday]
# sudo apt install scrot -y

The following packages were automatically installed and are no longer required:
crackmapexec          libgles1              libroc0.3
firebird3.0-common   libglusterfs0         libsuperlu6
firebird3.0-common-doc libglvnd-core-dev     libtag1v5
fonts-liberation2    libglvnd-dev          libtag1v5-vanilla
freerdp2-x11         libgspell-1-2        libtagc0
hydra-gtk            libgtk2.0-0t64       libu2f-udev
ibverbs-providers   libgtk2.0-bin        libwebRTC-audio-processing1
icu-devtools        libgtk2.0-common     libwinpr2-2t64
libabsl20230802     libgtksourceview-3.0-1 libzip4t64
libarmadillo12      libgtksourceview-3.0-common openjdk-17-jre
libassuan0          libgtksourceviewmm-3.0-0v5 perl-modules-5.38
libavfilter9        libgumbo2            python3-appdirs
libbfbio1           libhdf5-103-1t64     python3-diskcache
libboost-iostreams1.83.0 libhdf5-hl-100t64    python3-hatch-vcs
libboost-thread1.83.0 libibverbs1          python3-hatchling
libcapstone4        libcicu-dev          python3-jose
libcephfs2          libiniparser1        python3-mistune0
libconfig++9v5      libjim0.82t64       python3-pathspect
```

```
GNU nano 8.4          usb_alert.py *
msg['From'] = '24mcsdf056@student.rru.ac.in'
msg['To'] = '24mcsdf056@student.rru.ac.in'

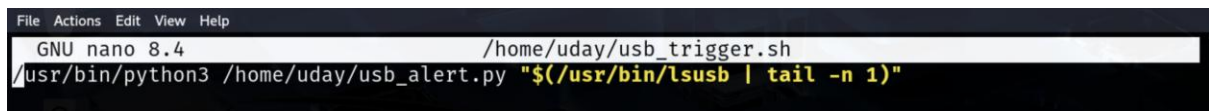
body = MIMEText(f"A USB device was inserted:\n\n{device_info}\n\nAllow it? [Yes/No]")
msg.attach(body)

with open(screenshot_path, 'rb') as file:
    part = MIMEBase('application', 'octet-stream')
    part.set_payload(file.read())
    encoders.encode_base64(part)
    part.add_header('Content-Disposition', f'attachment; filename="usb_screenshot.png"')
    msg.attach(part)

server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login('24mcsdf056@student.rru.ac.in', 'xxxx xxxx xxxx')
server.send_message(msg)
server.quit()
```

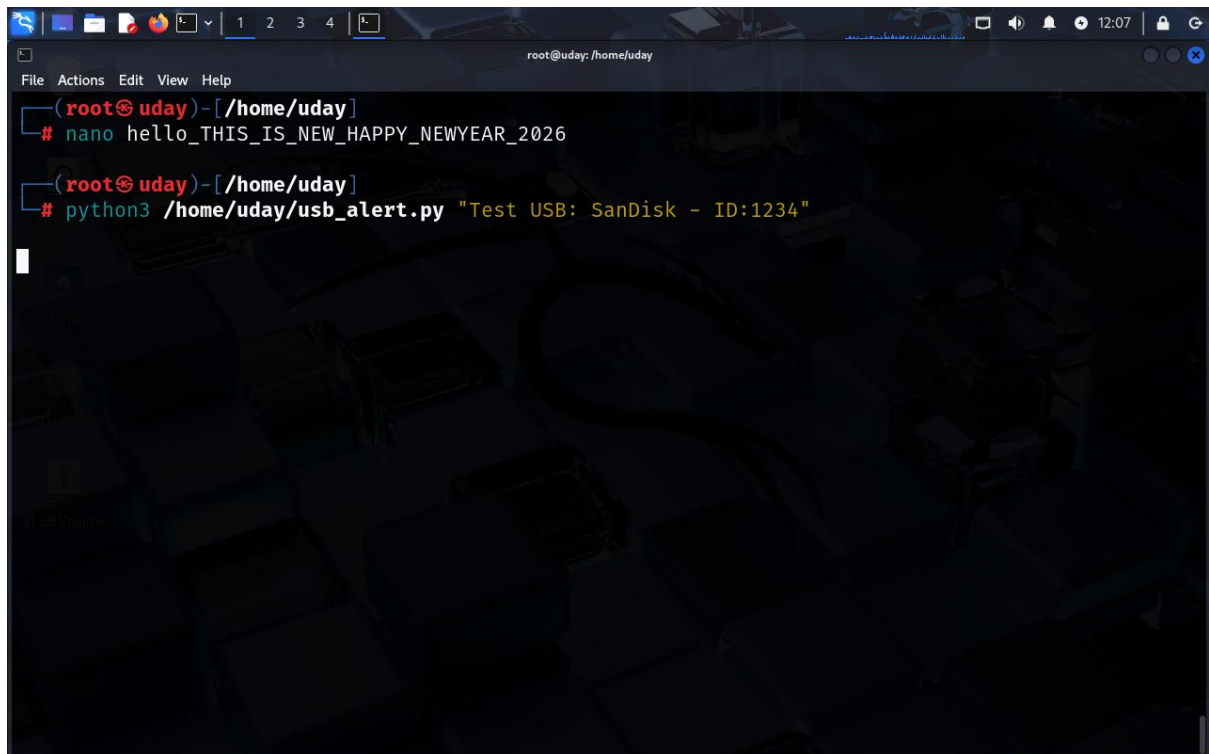
Here it the updated version code where it automated later by adding some lined above

Again, writing the small code to do automation task and providing the full permission

A terminal window with a dark background. At the top, a menu bar shows 'File Actions Edit View Help'. Below it, the text 'GNU nano 8.4' is on the left and '/home/uday/usb\_trigger.sh' is on the right. The main area shows a single line of code: '/usr/bin/python3 /home/uday/usb\_alert.py "\$(/usr/bin/lsusb | tail -n 1)"'.

```
File Actions Edit View Help
GNU nano 8.4 /home/uday/usb_trigger.sh
/usr/bin/python3 /home/uday/usb_alert.py "$(/usr/bin/lsusb | tail -n 1)"
```

Again, with message I also received the screenshot

A terminal window with a dark background. At the top, a menu bar shows 'File Actions Edit View Help'. Below it, the text 'root@uday: /home/uday' is on the right. The main area shows two commands being executed: '# nano hello\_THIS\_IS\_NEW\_HAPPY\_NEWYEAR\_2026' and '# python3 /home/uday/usb\_alert.py "Test USB: SanDisk - ID:1234"'.

```
File Actions Edit View Help
root@uday: /home/uday
(root@uday) - [/home/uday]
# nano hello_THIS_IS_NEW_HAPPY_NEWYEAR_2026
(root@uday) - [/home/uday]
# python3 /home/uday/usb_alert.py "Test USB: SanDisk - ID:1234"
```

### Screenshot Capture

When a USB device is inserted, the system immediately takes a screenshot using the scrot tool to capture the screen state.

### Email with Attachment

The email alert now includes the screenshot as an attachment, along with the USB details, providing visual evidence of what was on the screen when the device was plugged in.

Form this we have more **Benefit**

This addition increases the credibility and forensic value of alerts. It allows the recipient to **see who was using the system or what was running on the screen**, which enhances monitoring and auditing.

12:09

77



## USB Alert - Permission Required



Inbox



24mcsdf056@stud... 12:08 pm

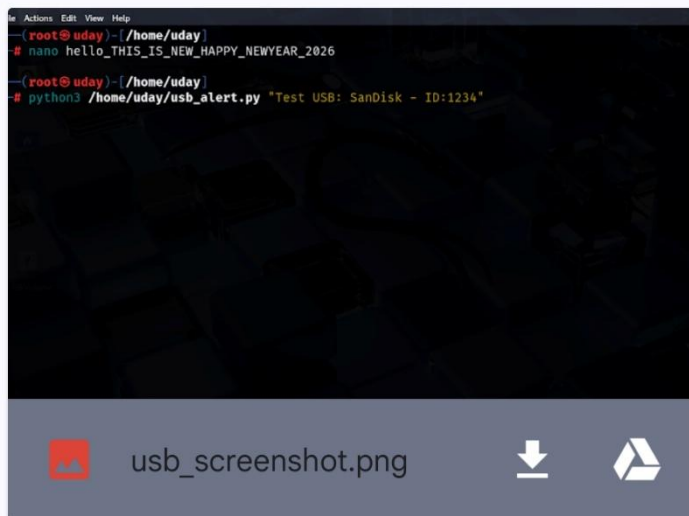


to me ▾

A USB device was inserted:

Test USB: SanDisk - ID:1234

Allow it? [Yes/No]



↩ Reply

↩↩ Reply all

➦ Forward

Available add-ons:





## Advantages

This project offers several key advantages. It does **not require any external hardware**, making it cost-effective and easy to deploy. It is fully compatible with **any Linux system**, allowing broad applicability across different environments. The system provides **real-time email alerts** when a USB device is connected, ensuring immediate awareness of any physical access attempts. Additionally, it uses **very low system resources**, making it suitable even for low-end machines. Most importantly, the solution is **highly customizable**, allowing for easy integration of future improvements such as logging, whitelisting, or GUI enhancements.

Also, in very simpler words This project, *SniffUSB*, was designed to monitor unauthorized USB insertions in real-time and alert the system owner via email, including a live screenshot for verification. The system effectively combines Python scripting, Linux automation, and security monitoring tools like udevadm and scrot.

Through this project, I gained practical experience in:

- Detecting hardware events using udev rules.
- Sending email alerts with device information and screenshots.
- Automating security responses using Python and Linux tools.

The alert system enhances physical security for machines by immediately notifying the user and seeking permission when any USB device is connected. It can be particularly useful in environments where physical access is a threat vector (e.g., public labs, server rooms, or shared workspaces).

This lightweight solution requires no investment and runs on any Linux system, making it ideal for real-time USB monitoring and email-based alerts, even in resource-constrained settings.

## Conclusion

This project helped me understand how a simple Python script can be used to improve real-time security monitoring on a Linux system. By automatically detecting USB device insertions and sending an email alert with detailed information and a screenshot, I was able to create a useful and practical security tool. The project taught me how to use tools like udevadm, scrot, and Python's email libraries in a real-world scenario. I also learned how to automate actions based on system events, which can be useful in many cybersecurity tasks. Overall, I feel more confident in writing scripts that can make a system more secure and responsive.