```
import pyudev
import time
import sys
import os

# ================================================================================
# PROJECT 2: USB DEVICE CONTROL TOOLKIT
# AUTHOR: [Your Name]
# PURPOSE: Detects USB Mass Storage devices and enforces a Whitelist policy.
# ================================================================================

# --- CONFIGURATION ---
# REPLACE with your actual device Serial Number found during testing.
USB_WHITELIST = ["9PGYG6H6WCNNS8TO", "ANOTHER_TRUSTED_SERIAL"]

def log_event(message):
    """
    Logs events to both the terminal and a local log file for auditing.
    """
    timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
    formatted_msg = f"[{timestamp}] {message}"

    print(formatted_msg)

    # Append to log file (Deliverable 3 Evidence)
    with open("usb_security.log", "a") as f:
        f.write(formatted_msg + "\n")

def block_device(device):
    """
    Disables the USB device by writing '0' to the kernel 'authorized' interface.
    This effectively cuts power/data to the specific USB port.
    """
    try:
        # The 'authorized' file controls if the OS allows the device to communicate
        auth_path = os.path.join(device.sys_path, 'authorized')

        if os.path.exists(auth_path):
            with open(auth_path, 'w') as f:
                f.write('0') # 0 = Disabled, 1 = Enabled

            log_event(f"!!! ACTION TAKEN: BLOCKED UNAUTHORIZED DEVICE:
{device.get('ID_MODEL')} !!!")
            print(f"    [+] Path: {device.sys_path}")
            print("    [+] Status: Port Disabled via Kernel.")
        else:
            log_event(f"[-] Warning: Could not find 'authorized' control file for
{device.get('ID_MODEL')}")

    except Exception as e:
```

```
        log_event(f"[!] Error attempting to block: {str(e)}")

def monitor_usb():
    """
    Main loop that listens for Kernel Udev events.
    """
    context = pyudev.Context()
    monitor = pyudev.Monitor.from_netlink(context)

    # Filter only for 'usb' subsystem events
    monitor.filter_by(subsystem='usb')

    print("================================================")
    print("[*] USB FIREWALL ACTIVE - PROTECTING SYSTEM")
    print(f"[*] Whitelisted Devices: {len(USB_WHITELIST)}")
    print("================================================")

    # Continuous listening loop
    for device in iter(monitor.poll, None):
        if device.action == 'add':
            # Wait briefly for attributes to initialize
            time.sleep(1)

            # --- INPUT SANITIZATION ---
            # Get the raw serial and remove any hidden non-alphanumeric characters
            # This fixes issues with hidden null bytes or spacing
            raw_serial = device.get('ID_SERIAL_SHORT', 'Unknown')
            serial = ''.join(char for char in str(raw_serial) if char.isalnum())
            model = device.get('ID_MODEL', 'Unknown')

            # Ignore internal USB hubs/interfaces, focus on the device itself
            if device.device_type == 'usb_interface':
                continue

            # --- DECISION ENGINE ---
            if serial in USB_WHITELIST:
                log_event(f"[+] ALLOWED: Trusted Device Connected - {model}
(Serial: {serial})")
            else:
                log_event(f"[!] ALERT: Untrusted Device Detected - {model} (Serial:
{serial})")
                block_device(device)

        elif device.action == 'remove':
             model = device.get('ID_MODEL', 'Unknown')
             print(f"[-] Device Removed: {model}")

if __name__ == "__main__":
    # Script must be run as Root to access /sys/ files for blocking
    if os.geteuid() != 0:
```

```
    print("[-] ERROR: This script requires ROOT privileges.")
    print("    Run with: sudo python3 usb_manager.py")
    sys.exit(1)

try:
    monitor_usb()
except KeyboardInterrupt:
    print("\n[*] Stopping USB Firewall...")
    sys.exit()
```