```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import pickle

# Step 1: Load or generate dataset
# If you already generated it, use: df = pd.read_csv("synthetic_keylogger_intrusion_data.csv")

# Otherwise, regenerate it
def generate_entry():
    suspicious_cmds = ["whoami", "ipconfig", "tasklist", "netstat", "msfvenom", "exploit", "reverse_shell", "ncat", "powershell"]
    normal_cmds = ["hello", "email", "meeting", "project", "thanks", "assignment", "notes", "login", "password123"]
    is_malicious = np.random.rand() < 0.3

    if is_malicious:
        cmd_count = np.random.randint(1, 4)
        keystrokes = " && ".join(np.random.choice(suspicious_cmds, size=cmd_count))
        suspicious_keywords = sum(1 for word in suspicious_cmds if word in keystrokes)
        access_to_admin = 1
        repeated_password_inputs = np.random.randint(2, 6)
    else:
        cmd_count = np.random.randint(3, 6)
        keystrokes = " ".join(np.random.choice(normal_cmds, size=cmd_count))
        suspicious_keywords = 0
        access_to_admin = 0
        repeated_password_inputs = np.random.randint(0, 2)

    return {
        "keystroke": keystrokes,
        "keystroke_length": len(keystrokes),
        "command_detected": int(any(cmd in keystrokes for cmd in suspicious_cmds)),
        "repeated_password_inputs": repeated_password_inputs,
        "suspicious_keywords": suspicious_keywords,
        "access_to_admin_tools": access_to_admin,
        "hour_of_day": np.random.randint(0, 24),
        "label": int(is_malicious)
    }

df = pd.DataFrame([generate_entry() for _ in range(500)])

# Step 2: Feature selection
features = ["keystroke_length", "command_detected", "repeated_password_inputs",
            "suspicious_keywords", "access_to_admin_tools", "hour_of_day"]
X = df[features]
y = df["label"]

# Step 3: Split and train model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 4: Evaluation
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))

# Step 5: Save model to .pkl
with open("keylogger_intrusion_model.pkl", "wb") as f:
    pickle.dump(model, f)

print("✅ Model saved as 'keylogger_intrusion_model.pkl'")
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        62
           1       1.00      1.00      1.00        38

    accuracy                           1.00       100
   macro avg       1.00      1.00      1.00       100
weighted avg       1.00      1.00      1.00       100

✅ Model saved as 'keylogger_intrusion_model.pkl'
```

```
import re
import pickle
from datetime import datetime

# ✅ Load the trained model (ensure file is uploaded in /content/)
model_path = "/content/keylogger_intrusion_model.pkl"
keylog_path = "/content/normal_keylog.txt"
```

```
# Load model
with open(model_path, "rb") as f:
    model = pickle.load(f)

# Read keylog data
with open(keylog_path, "r") as f:
    raw_data = f.read().lower()

# ✅ Feature extraction function with 6 features
def extract_features_from_keylog(raw_text):
    special_keys = ['enter', 'ctrl', 'alt', 'esc', 'shift', 'tab', 'backspace']
    hacking_keywords = ['netstat', 'msfvenom', 'exploit', 'reverse', 'shell', 'payload', 'tasklist']

    keystroke_length = len(re.findall(r"[a-z0-9]", raw_text))  # Count of normal characters
    special_keys_count = sum(raw_text.count(k) for k in special_keys)
    command_count = sum(raw_text.count(cmd) for cmd in ['netstat', 'tasklist', 'whoami', 'ipconfig'])
    contains_keywords = sum(raw_text.count(k) for k in hacking_keywords)
    password_attempts = raw_text.count("password")
    hour_of_day = datetime.now().hour  # Current hour (0-23)

    # ✅ Ensure feature order matches training:
    # ["keystroke_length", "command_detected", "repeated_password_inputs",
    #  "suspicious_keywords", "access_to_admin_tools", "hour_of_day"]
    return [[
        keystroke_length,
        command_count,
        password_attempts,
        contains_keywords,
        special_keys_count,
        hour_of_day
    ]]

# Extract features
features = extract_features_from_keylog(raw_data)

# ✅ Make prediction
prediction = model.predict(features)[0]

# ✅ Output result
if prediction == 1:
    print("⚠ Potential Exploitation Detected")
else:
    print("✅ Normal Activity")
```

```
✅ Normal Activity
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Ranc
  warnings.warn(
```

Generate   create a dataframe with 2 columns and 10 rows   Close

```
import re
import pickle

# ✅ Load the trained model
model_path = "/content/keylogger_intrusion_model.pkl"
with open(model_path, "rb") as f:
    model = pickle.load(f)

# ✅ Feature extractor
def extract_features_from_keylog(raw_text):
    special_keys = ['enter', 'ctrl', 'alt', 'esc', 'shift', 'tab', 'backspace']
    hacking_keywords = ['netstat', 'msfvenom', 'exploit', 'reverse', 'shell', 'payload', 'tasklist']

    keystroke_length = len(re.findall(r"[a-z0-9]", raw_text))  # Count of normal characters
    special_keys_count = sum(raw_text.count(k) for k in special_keys)
    command_count = sum(raw_text.count(cmd) for cmd in ['netstat', 'tasklist', 'whoami', 'ipconfig'])
    contains_keywords = sum(raw_text.count(k) for k in hacking_keywords)
    password_attempts = raw_text.count("password")
    hour_of_day = 14  # You can modify this if you want dynamic time-based simulation

    return [[keystroke_length, command_count, password_attempts, contains_keywords, special_keys_count, hour_of_day]]

# ✅ List of test keylog files
log_files = ["normal_keylog.txt", "malicious_keylog.txt"]

# ✅ Run tests
for file in log_files:
    try:
        with open(f"/content/{file}", "r") as f:
            raw_data = f.read().lower()

        features = extract_features_from_keylog(raw_data)
        prediction = model.predict(features)[0]

        print(f"\n📄 Testing: {file}")
        print("🔍 Prediction:", "⚠ Potential Exploitation Detected" if prediction == 1 else "✅ Normal Activity")
    except FileNotFoundError:
        print(f"❌ File not found: {file}")
```

```
📄 Testing: normal_keylog.txt
🔍 Prediction: ✅ Normal Activity

📄 Testing: malicious_keylog.txt
🔍 Prediction: ⚠ Potential Exploitation Detected
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Ranc
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Ranc
  warnings.warn(
```