```python

import hashlib
import os
import json
def calculate_file_hash(file_path):
    sha256 = hashlib.sha256()
    try:
        with open(file_path, 'rb') as f:
            while chunk := f.read(8192):
                sha256.update(chunk)
        return sha256.hexdigest()
    except Exception as e:
        print(f"[ERROR] Cannot read '{file_path}': {e}")
        return None
def save_file_hashes(directory, output_file):
    file_hashes = {}
    for root, _, files in os.walk(directory):
        for file in files:
            full_path = os.path.join(root, file)
            hash_val = calculate_file_hash(full_path)
            if hash_val:
                file_hashes[full_path] = hash_val

    print(f"[DEBUG] Scanned {len(file_hashes)} files.")

    if file_hashes:
```

Output Python v3.12.7 Pyodide 0.27.3

```
[WARNING] No files found to hash.
```

11:11:02 AM

```
[WARNING] No files found to hash.
```

11:11:51 AM

```
[DEBUG] Scanned 0 files.
[WARNING] No files found to hash. Check your folder path!
```

```
        with open(output_file, 'w') as f:
            json.dump(file_hashes, f, indent=4)
        print(f"[INFO] {len(file_hashes)} file hashes saved to '{output_file}'")
    else:
        print("[WARNING] No files found to hash. Check your folder path!")
def check_file_changes(directory, hash_file):
    if not os.path.exists(hash_file):
        print("[ERROR] Hash record not found. Run save_file_hashes() first.")
        return

    with open(hash_file, 'r') as f:
        old_hashes = json.load(f)

    current_hashes = {}
    for root, _, files in os.walk(directory):
        for file in files:
            full_path = os.path.join(root, file)
            current_hashes[full_path] = calculate_file_hash(full_path)
    all_paths = set(old_hashes) | set(current_hashes)
    for path in sorted(all_paths):
        old = old_hashes.get(path)
        new = current_hashes.get(path)
        if old is None:
            print(f"[NEW]      {path}")
        elif new is None:
            print(f"[DELETED]  {path}")
```

tput Python v3.12.7 Pyodide 0.27.3

```
ARNING] No files found to hash.
```

1:02 AM

```
ARNING] No files found to hash.
```

1:51 AM

```
EBUG] Scanned 0 files.
ARNING] No files found to hash. Check your folder path!
```

```
    with open(hash_file, 'r') as f:
        old_hashes = json.load(f)

    current_hashes = {}
    for root, _, files in os.walk(directory):
        for file in files:
            full_path = os.path.join(root, file)
            current_hashes[full_path] = calculate_file_hash(full_path)
    all_paths = set(old_hashes) | set(current_hashes)
    for path in sorted(all_paths):
        old = old_hashes.get(path)
        new = current_hashes.get(path)
        if old is None:
            print(f"[NEW]      {path}")
        elif new is None:
            print(f"[DELETED]  {path}")
        elif old != new:
            print(f"[CHANGED]  {path}")
        else:
            print(f"[UNCHANGED]{path}")
if __name__ == "__main__":
    folder_to_monitor = "C:/Users/Deepa/Documents/project_files"
    hash_log_file = "file_hashes.json"
    save_file_hashes(folder_to_monitor, hash_log_file)
```

```python
import requests
from bs4 import BeautifulSoup
import urllib.parse
sql_payloads = ["' OR 1=1--", "' AND 'a'='a", "'; DROP TABLE users; --"]
xss_payloads = ['<script>alert("XSS")</script>', '" onmouseover="alert(1)"']
def get_all_forms(url):
    try:
        res = requests.get(url, timeout=10)
        soup = BeautifulSoup(res.content, "html.parser")
        return soup.find_all("form")
    except Exception as e:
        print(f"[ERROR] Failed to load page: {e}")
        return []
def get_form_details(form):
    details = {"action": form.get("action"), "method": form.get("method", "get").lower(), "in
    for input_tag in form.find_all(["input", "textarea"]):
        name = input_tag.get("name")
        if name:
            details["inputs"].append(name)
    return details
def submit_form(form_details, url, payload):
    target = urllib.parse.urljoin(url, form_details["action"] or "")
    data = {input_name: payload for input_name in form_details["inputs"]}
    try:
        if form_details["method"] == "post":
```

ut Python v3.12.7 Pyodide 0.27.3

:34 AM

utput or return value

```
            return requests.post(target, data=data, timeout=10)
        else:
            return requests.get(target, params=data, timeout=10)
    except Exception as e:
        print(f"[ERROR] Failed to submit form: {e}")
        return None
f scan_url(url):
    forms = get_all_forms(url)
    print(f"[INFO] Found {len(forms)} form(s) at {url}\n")

    for i, form in enumerate(forms):
        details = get_form_details(form)
        print(f"[FORM {i+1}] Method: {details['method'].upper()}, Action: {details['action']}")

        for payload in sql_payloads + xss_payloads:
            print(f"  [TEST] Trying payload: {payload}")
            response = submit_form(details, url, payload)
            if response and payload.lower() in response.text.lower():
                if "<script" in payload.lower():
                    print("  [ALERT] Possible XSS vulnerability detected!")
                elif "'" in payload or "--" in payload:
                    print("  [ALERT] Possible SQL Injection vulnerability detected!")
__name__ == "__main__":
    target_url = "http://localhost:8000"

    scan_url(target_url)
```

Python v3.12.7 Pyodide 0.27.3

AM

ut or return value

```python
port scanner module
import socket
def scan_ports(host, ports):
    print(f"[+] Scanning {host}...")
    for port in ports:
        try:
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
                sock.settimeout(0.5)
                result = sock.connect_ex((host, port))
                if result == 0:
                    print(f"  [OPEN] Port {port}")
        except Exception as e:
            print(f"  [ERROR] {port}: {e}")
```

```
Brute Forcer Module
import requests
def brute_force_login(url, username_field, password_field, usernames, passwords):
    for user in usernames:
        for pwd in passwords:
            data = {username_field: user, password_field: pwd}
            res = requests.post(url, data=data)
            if "Welcome" in res.text or res.status_code == 200:
                print(f"[SUCCESS] Username: {user}, Password: {pwd}")
                return
            else:
                print(f"[FAILED] {user}:{pwd}")
```

Entry Script

```python
from core.scanner import scan_ports
from core.brute_forcer import brute_force_login
def main():
    print("=== Penetration Testing Toolkit ===")
    host = "127.0.0.1"
    ports = [21, 22, 80, 443]
    scan_ports(host, ports)
    url = "http://127.0.0.1/login"
    brute_force_login(
        url,
        username_field="user",
        password_field="pass",
        usernames=["admin", "test"],
        passwords=["1234", "password", "admin"]
    )
if __name__ == "__main__":
    main()
```

```
Documentation

- **Port Scanner**: Scans TCP ports on target host.
- **Brute Forcer**: Attempts login combinations using POST forms.
```bash
python main.py
```
```

```python
task 4
import os
import tkinter as tk
from tkinter import filedialog, messagebox
from cryptography.fernet import Fernet
import base64
import hashlib
def generate_key(password):
    return base64.urlsafe_b64encode(hashlib.sha256(password.encode()).digest())
def encrypt_file(file_path, password):
    key = generate_key(password)
    fernet = Fernet(key)
    try:
        with open(file_path, 'rb') as f:
            data = f.read()
        encrypted = fernet.encrypt(data)
        with open(file_path + ".enc", 'wb') as f:
            f.write(encrypted)
        messagebox.showinfo("Success", f"Encrypted: {file_path}.enc")
    except Exception as e:
        messagebox.showerror("Error", f"Encryption failed: {e}")

def decrypt_file(file_path, password):
    key = generate_key(password)
    fernet = Fernet(key)
    try:
```

```
        with open(file_path, 'rb') as f:
            data = f.read()
        encrypted = fernet.encrypt(data)
        with open(file_path + ".enc", 'wb') as f:
            f.write(encrypted)
        messagebox.showinfo("Success", f"Encrypted: {file_path}.enc")
    except Exception as e:
        messagebox.showerror("Error", f"Encryption failed: {e}")

def decrypt_file(file_path, password):
    key = generate_key(password)
    fernet = Fernet(key)
    try:
        with open(file_path, 'rb') as f:
            encrypted = f.read()
        decrypted = fernet.decrypt(encrypted)
        original_path = file_path.replace(".enc", "_decrypted")
        with open(original_path, 'wb') as f:
            f.write(decrypted)
        messagebox.showinfo("Success", f"Decrypted: {original_path}")
    except Exception as e:
        messagebox.showerror("Error", f"Decryption failed: {e}")
def build_gui():
    def choose_encrypt():
        path = filedialog.askopenfilename()
        if path:
```

put Python v3.12.7 Pyodide 0.27.3

```
        '
            pwd = password_entry.get()
            if pwd:
                encrypt_file(path, pwd)
            else:
                messagebox.showwarning("Input Needed", "Please enter a password!")
    def choose_decrypt():
        path = filedialog.askopenfilename()
        if path:
            pwd = password_entry.get()
            if pwd:
                decrypt_file(path, pwd)
            else:
                messagebox.showwarning("Input Needed", "Please enter a password!")
    root = tk.Tk()
    root.title("AES-256 File Encryptor")
    root.geometry("400x200")
    root.resizable(False, False)
    tk.Label(root, text="🔒 Enter Password:", font=("Arial", 12)).pack(pady=10)
    password_entry = tk.Entry(root, show="*", width=30)
    password_entry.pack()
    tk.Button(root, text="Encrypt File", width=20, command=choose_encrypt).pack(pady=10)
    tk.Button(root, text="Decrypt File", width=20, command=choose_decrypt).pack()
    root.mainloop()
if __name__ == "__main__":
    build_gui()
```