Security Hacker Tool - Documentation

This document provides a detailed overview of the Security Hacker Tool developed using Python Flask. It includes the application code, technical documentation, and a flowchart explanation.

Application Code

Flask Application Code (app.py)

```
from flask import Flask, request, render_template, redirect, url_for, jsonify
import os
import hashlib
import subprocess
import shutil
import magic

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads'
TOOLS_FOLDER = 'tools'
EXTRACTED_FOLDER = 'extracted'

os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(EXTRACTED_FOLDER, exist_ok=True)
```

Security Hacker Tool - Documentation

```
@app.route('/')

def index():

    frameworks = ['Cocos', 'Flutter', 'React Native', 'Cordova', 'Unity']

       actions = ['reflutter app', 'flutter dart objects dump', 'libil2cpp dumper', 'dump using Hermes

decompiler', 'cocos file dumper']

    return render_template('index.html', frameworks=frameworks, actions=actions)



@app.route('/process', methods=['POST'])

def process_file():

    try:

        # Step 1: File Upload and Framework Selection

        file = request.files['file']

        framework = request.form['framework']

        action = request.form['action']

        ip_address = request.form.get('ip_address', '')



        # Step 2: File Validation

        file_signature = magic.Magic(mime=True)

        file_type = file_signature.from_buffer(file.read(2048))

        file.seek(0)

        if not file.filename.endswith('.apk') or file_type != 'application/vnd.android.package-archive':

            return jsonify({'error': 'Invalid file format. Only APK files are allowed.'}), 400



        # Create a unique folder using SHA256 hash

        file_hash = hashlib.sha256(file.filename.encode()).hexdigest()
```

```
        extracted_path = os.path.join(EXTRACTED_FOLDER, file_hash)

        os.makedirs(extracted_path, exist_ok=True)



        # Save the file to the upload folder

        file_path = os.path.join(UPLOAD_FOLDER, file.filename)

        file.save(file_path)



        # Step 3: APK Extraction

        apk
```

Technical Documentation

1. App Structure:

- app.py: Main application logic.
- templates/: HTML templates folder.
  - index.html: Main UI for file upload and selection.
  - success.html: Output page after action execution.
- uploads/: Stores uploaded APK/Zip files.
- extracted/: Stores APK extracted files.
- tools/: Contains all required tools.

2. Key Features:

- File validation using MIME types.
- Extraction using apktool.
- Actions executed based on user selection and file content.

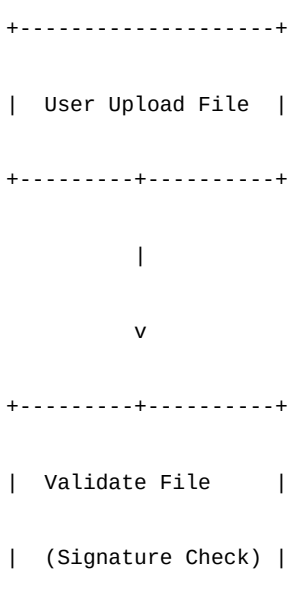
- Error handling with descriptive error messages.

3. Dependencies:

- Flask
- Python 3
- apktool
- python-magic library for MIME type detection.

4. Flowchart Explanation:

- Input Section: Upload APK and dropdown selections.
- Processing Section:
  - Validate file type.
  - Extract APK with apktool.
  - Execute selected action.
- Output Section: Display logs and paths.



Security Hacker Tool - Documentation