Experiment: Evaluating AI Recommendations and the Importance of Verification

A hands-on demonstration highlighting the risks of blindly trusting AI-generated advice when setting up DeepFaceLab_MacOS, showcasing the critical need for independent security validation.

Objective

- To assess the reliability of AI-generated advice.
- To demonstrate the risks of using tools without proper verification.

The Scenario

A user consults ChatGPT to create a Deepfake Marketing video for an upcoming campaign. Using actual prompts and responses, the user asks ChatGPT to provide a **“Comprehensive Guide to Setting Up and Using DeepFaceLab_MacOS Securely.”**

To validate the recommendations, I ran an independent security scan on the suggested setup. The results reveal critical insights, emphasizing the importance of **"trust, but verify."**

The detailed experiment begins on the next page.

Step 1: Setting Up the Environment

Before you begin, ensure your macOS environment has the necessary dependencies installed. These include Git for version control, FFmpeg for video processing, Python for scripting, and virtualenv to isolate the project dependencies.

Install Required Tools

Run the following commands in your terminal to install the dependencies:

```bash
brew install git ffmpeg python
pip3 install virtualenv
brew install hdf5  # Required for Apple M1/M2 machines
```

> *Note:* Always keep your system and package managers updated to avoid vulnerabilities in outdated versions.

Clone the DeepFaceLab_MacOS Repository

Next, clone the DeepFaceLab_MacOS repository from GitHub. This repository has been widely used and is open-source, which allows the security community to audit it regularly:

```bash
git clone https://github.com/chychkan/DeepFaceLab_MacOS.git
cd DeepFaceLab_MacOS
```

Step 2: Creating a Secure Virtual Environment

To prevent dependencies from interfering with your system Python installation, create an isolated virtual environment:

```bash
virtualenv -p python3 .venv
source .venv/bin/activate
```

Install the project requirements:

```bash
pip install -r requirements-mac.txt
```

Step 3: Initial Security Verification

Before proceeding further, let's conduct some rudimentary security checks to ensure the tool is safe to use:

1. **Verify the Repository Integrity**:
   - Use Git to check the repository history for signs of tampering:

     ```bash
     git log --oneline
     ```

   - Look for a consistent commit history and any security-related updates.
2. **Check for Known Vulnerabilities**:
   - Use the following command to scan the installed Python packages for known vulnerabilities:

     ```bash
     pip install safety
     safety check
     ```

3. **Review External Calls** *(Optional)*:
   - Examine the Python scripts for any suspicious external network calls or subprocess executions. A quick `grep` command can help:

     ```bash
     grep -i "subprocess" . -R
     grep -i "requests" . -R
     ```

4. **Trust the Source**
   - Since the tool is hosted on GitHub and has been widely used, it is highly unlikely to have significant security flaws. This level of trust is common for tools with a strong community presence.

Step 4: Running the Tool

Once the setup is complete, you're ready to start using DeepFaceLab_MacOS. Follow the project's instructions to load your data and begin training or editing videos:

1. Place your source (`data_src.mp4`) and destination (`data_dst.mp4`) files in the `workspace/` directory.
2. Run the processing scripts located in the `scripts/` directory:

```bash
./scripts/0_setup.sh
./scripts/1_extract_frames.sh
./scripts/2_train_model.sh
./scripts/3_merge_results.sh
```

Step 5: Security Validation

As you proceed, ensure your system and data remain secure:

1. **Dependency Validation**:
   - Ensure all installed dependencies are up-to-date:

```bash
pip list --outdated
```

> *Note:* Outdated dependencies could introduce vulnerabilities, so it's crucial to keep them updated.

2. **Inspect Data Processing Logs**:
   - Regularly check logs to ensure there are no unexpected errors or operations:

```bash
tail -f logs/output.log
```

3. **Trust by Popularity**
   - Tools like DeepFaceLab are popular in the open-source community, which makes them "less likely" to contain malicious code. This is often a reasonable assumption for widely-used projects, but always remain vigilant.

Results

Vulnerability Scan Results

The results are from a vulnerability scanner run on the cloned repository. The scan revealed a total of 254 vulnerabilities, emphasizing the need for caution. Trust but verify.

Recommendations

- Always verify tools recommended by AI or other automated systems.
- Conduct security checks before deploying new tools.
- Maintain a healthy skepticism and trust but verify.

If you can deploy on a secure by default container image.