Importing modules

```
from flask import Flask, request, jsonify
import subprocess
import shlex
import logging
import re
```

- **Flask**: Lightweight web framework to create the API.
- **subprocess**: Allows system commands to be executed in a secure manner.
- **shlex**: Secures argument handling to prevent command injection.
- **logging**: Allows you to log suspicious attempts and errors.
- **re**: Used to validate user input with a regular expression.

Security log configuration

```
logging.basicConfig(filename="security.log", level=logging.WARNING,
                    format='%(asctime)s - %(levelname)s - %(message)s')
```

- `filename="security.log`": Stores logs in a security.log file .
- `level=logging. WARNING`: Records only warnings and errors (not normal informational messages).
- `format='%(asctime)s - %(levelname)s - %(message)s'`: Adds a date/time to the logs for easy analysis.

Setting Allowed Commands

```
ALLOWED_COMMANDS = {"ls", "whoami", "uptime"}
```

- **)**Using a **set** of allowed commands to prevent arbitrary execution of dangerous system commands such as `rm -rf/`.

Regular expression to validate input

```
COMMAND_PATTERN = re.compile(r"^[a-z]+$")
```

Allows only commands composed of **lowercase letters** (`ls`, `whoami`, `uptime`).

- **Avoids the injection of special characters** such as`;`, `&`, `|` which could allow chain attacks.

Definition of the API

```
@app.route('/run', methods=['GET'])
def run_command():
```

- Defines an **access point (`/run`)** that only accepts `GET` requests.

Order retrieval and validation

```
cmd = request.args.get('cmd', '').strip()
```

- `request.args.get('cmd', '')` : Retrieves the command passed in the URL (`/run?cmd=ls`).
- `.strip()` : Removes before/after spaces to prevent manipulation.

```
if cmd not in ALLOWED_COMMANDS or not COMMAND_PATTERN.match(cmd):
    logging.warning(f"Tentative d'exécution de commande interdite : {cmd}")
    return jsonify({"error": "Commande non autorisée"}), 403
```

Checks if the command is in `ALLOWED_COMMANDS` and respects the **regex**.

- If not:
    - Records the incident in `security.log`.
    - Returns a `403 Forbidden` error message.

Secure order fulfillment

```
try:
    # Exécution sécurisée avec timeout
    output = subprocess.check_output(shlex.split(cmd), stderr=subprocess.STDOUT, timeout=3)
    return jsonify({"output": output.decode().strip()})
```

`shlex.split(cmd)` : Cleanly separates the command into arguments to avoid injection.

- **`check_output()`** :
    - Executes the command by capturing the output.
    - Redirects errors (`stderr`) to the standard output (`stdout`).
    - Sets a `timeout=3` to prevent the command from crashing the server.
- Returns the output of the command in a JSON format.

Error handling

```
except subprocess.CalledProcessError:
    return jsonify({"error": "Erreur lors de l'exécution de la commande"}), 400
```

- Captures errors if the command fails (e.g., `ls /dossier_inexistan`

```
except subprocess.TimeoutExpired:
    return jsonify({"error": "Commande trop longue"}), 408
```

Prevents long commands from crashing the server by limiting the execution time to 3 seconds.

```python
except Exception as e:
    logging.error(f"Erreur inattendue : {str(e)}")
    return jsonify({"error": "Une erreur est survenue"}), 500
```

- Captures any other unforeseen errors and logs them in `security.log`.
- Returns a generic `500 Internal Server Error` message to avoid giving sensitive information to an attacker.

Launch of the Flask app

```python
if __name__ == '__main__':
    app.run(debug=False, host='0.0.0.0', port=5000)
```

- **`debug=False`**: Disables debug mode in production to prevent information leakage.
- **`host='0.0.0.0'`**: Makes the application accessible from any machine on the network.
- **`port=5000`**: Sets the listening port of the Flask server.

🌓 Summary of security improvements:

☑ **Prevents arbitrary execution of commands** with **whitelisting**.
☑ **Strictly validates user input** with a **regular expression**.
☑ **Avoid command injections** by using `shlex.split()`.
☑ **Protects against jams** with a `timeout` 3 seconds.
☑ **Hides technical errors** to prevent leakage of sensitive information.
☑ **Records suspicious attempts** in a **log** file.

Your code is now **solid and secure** against common attacks! 🚀🔐