

# AgenticCyberWorkflow

=====

## Overview

-----

AgenticCyberWorkflow is an autonomous cybersecurity pipeline built with LangGraph, LangChain, and Streamlit. The project simulates dynamic security scans using tools such as nmap, gobuster, ffuf, and sqlmap by converting high-level security instructions into a series of sequential, executable tasks. With robust state management via Pydantic, automated task execution, and a user-friendly web interface for real-time monitoring, this project lays the foundation for scalable and modular security testing and audit automation.

## Features

-----

- **Automated Task Decomposition:** Converts high-level security instructions into discrete, actionable scan tasks.
- **Scope Enforcement:** Fully functional domain/IP filtering ensures that all scans stay within the defined boundaries.
- **Dynamic Task Management:** Automatically updates the task list based on intermediate scan results.
- **Error Handling & Retries:** Automatically handles task failures with retries and alternate strategies.

- **Real-Time Monitoring:** Integrated Streamlit UI provides live updates of scan outputs and task statuses.
- **Detailed Logging:** Comprehensive logging for auditing and troubleshooting.

## Architecture

-----

### 1. **Task Decomposition & Workflow Generation:**

Utilizes LangChain to parse high-level instructions and break them into a series of actionable tasks.

### 2. **Agentic Task Scheduler & Executor:**

Leverages LangGraph to build a dynamic task graph that manages sequential execution and handles task failures with retries.

### 3. **Scope Enforcement Module:**

Validates that all scans adhere to user-defined domains or IP ranges, ensuring that operations remain within permitted boundaries.

### 4. **State Management:**

Uses a Pydantic model (SecurityScanState) to manage and validate the state across the workflow, maintaining consistency throughout the scanning process.

## 5. **\*\*User Interface:\*\***

A Streamlit-based dashboard visualizes ongoing scan outputs, task statuses, and logs in real time.

## Installation and Setup

-----

### **\*\*System Requirements:\*\***

- Python 3.11
- Poetry or pip for dependency management
- Security tools: Ensure that nmap, gobuster, ffuf, and sqlmap are installed and available in your system PATH.

### **\*\*Installation Steps:\*\***

#### 1. **\*\*Clone the Repository:\*\***

```
git clone
```

```
https://github.com/yourusername/AgenticCyberWorkflow.git
```

```
cd AgenticCyberWorkflow
```

```
mathematica
```

CopyEdit

## 2. **\*\*Set Up a Virtual Environment:\*\***

```
``bash
```

```
python -m venv venv
```

```
source venv/bin/activate # On Windows:  
venv\Scripts\activate
```

## 3. **Install Dependencies:** Using Poetry:

```
bash
```

CopyEdit

```
poetry install
```

Or using pip:

```
bash
```

CopyEdit

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

## 4. **Configuration:**

- Create a .env file if needed.
- Define environment variables such as allowed domains and IP ranges.
- Example:

```
ini
```

CopyEdit

ALLOWED\_DOMAINS=google.com,github.com

ALLOWED\_IP\_RANGES=192.168.1.0/24

## Usage

### Running the Application:

- **Command-Line Interface:** Start the agentic workflow with:

bash

CopyEdit

```
python run_agent.py --scope .env --instruction "Scan github.com for open ports and discover directories"
```

- **Streamlit UI:** Launch the interactive UI with:

bash

CopyEdit

```
streamlit run app.py
```

### How It Works:

- **Input:**  
The application accepts high-level instructions along with a scan scope configuration.
- **Workflow:**  
The system decomposes the instructions into sequential tasks:
  1. Execute an nmap scan.
  2. Run a gobuster scan based on nmap results.

3. Execute an ffuf scan.

4. Run a sqlmap scan.

- **Output:**

The results of each task are logged and displayed in the Streamlit UI.

## Testing & Validation

- **Unit Tests:**

Unit tests are implemented using Pytest, covering:

- Task execution flow.
- Scope enforcement logic.
- Error handling and retry mechanisms.

- **Running Tests:**

bash

CopyEdit

pytest tests/

## Demo & Documentation

- **Video Demo:**

A video demonstration is available that shows the setup, execution, and outputs of the pipeline.

- **Further Documentation:**

Detailed explanations and configuration options are available in the project's Wiki.

## Contributing

Contributions are welcome! Please fork the repository, make your changes, and submit a pull request. For major changes, please open an issue first to discuss your ideas.

## **License**

This project is licensed under the MIT License.

## **Contact**

For further information or questions, please contact [Your Name] at [your.email@example.com].

yaml

CopyEdit

---

Once you paste this content into Word, you can adjust formatting as needed and then save it as "README.docx".

o3-mini

## **You said:**

give me in proper file that i can copy

Reasoned for 21 seconds

Below is the complete README file content in a single block. Simply copy the text below, paste it into your preferred text editor (like Microsoft Word), and then save the document as "README.docx".

---

pgsql

CopyEdit

AgenticCyberWorkflow

=====

## Overview

-----

AgenticCyberWorkflow is an autonomous cybersecurity pipeline built with LangGraph, LangChain, and Streamlit. The project simulates dynamic security scans using tools such as nmap, gobuster, ffuf, and sqlmap by converting high-level security instructions into a series of sequential, executable tasks. With robust state management via Pydantic, automated task execution, and a user-friendly web interface for real-time monitoring, this project lays the foundation for scalable and modular security testing and audit automation.

## Features

-----

- **Automated Task Decomposition:** Converts high-level security instructions into discrete, actionable scan tasks.
- **Scope Enforcement:** Fully functional domain/IP filtering ensures that all scans stay within the defined boundaries.
- **Dynamic Task Management:** Automatically updates the task list based on intermediate scan results.



- **Error Handling & Retries:** Automatically handles task failures with retries and alternate strategies.
- **Real-Time Monitoring:** Integrated Streamlit UI provides live updates of scan outputs and task statuses.
- **Detailed Logging:** Comprehensive logging for auditing and troubleshooting.

## Architecture

-----

### 1. **Task Decomposition & Workflow Generation:**

Utilizes LangChain to parse high-level instructions and break them into a series of actionable tasks.

### 2. **Agentic Task Scheduler & Executor:**

Leverages LangGraph to build a dynamic task graph that manages sequential execution and handles task failures with retries.

### 3. **Scope Enforcement Module:**

Validates that all scans adhere to user-defined domains or IP ranges, ensuring that operations remain within permitted boundaries.

### 4. **State Management:**

Uses a Pydantic model (SecurityScanState) to manage and validate the state across the workflow, maintaining consistency throughout the scanning process.

## 5. **\*\*User Interface:\*\***

A Streamlit-based dashboard visualizes ongoing scan outputs, task statuses, and logs in real time.

## Installation and Setup

-----

### **\*\*System Requirements:\*\***

- Python 3.11
- Poetry or pip for dependency management
- Security tools: Ensure that nmap, gobuster, ffuf, and sqlmap are installed and available in your system PATH.

### **\*\*Installation Steps:\*\***

#### 1. **\*\*Clone the Repository:\*\***

```
git clone
```

```
https://github.com/yourusername/AgenticCyberWorkflow.git
```

```
cd AgenticCyberWorkflow
```

```
mathematica
```

CopyEdit

## 2. **\*\*Set Up a Virtual Environment:\*\***

```
``bash
```

```
python -m venv venv
```

```
source venv/bin/activate # On Windows:  
venv\Scripts\activate
```

## 3. **Install Dependencies:** Using Poetry:

```
bash
```

CopyEdit

```
poetry install
```

Or using pip:

```
bash
```

CopyEdit

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

## 4. **Configuration:**

- Create a .env file if needed.
- Define environment variables such as allowed domains and IP ranges.
- Example:

```
ini
```

CopyEdit

ALLOWED\_DOMAINS=google.com,github.com

ALLOWED\_IP\_RANGES=192.168.1.0/24

## Usage

### Running the Application:

- **Command-Line Interface:** Start the agentic workflow with:

bash

CopyEdit

```
python run_agent.py --scope .env --instruction "Scan github.com for open ports and discover directories"
```

- **Streamlit UI:** Launch the interactive UI with:

bash

CopyEdit

```
streamlit run app.py
```

### How It Works:

- **Input:**  
The application accepts high-level instructions along with a scan scope configuration.
- **Workflow:**  
The system decomposes the instructions into sequential tasks:
  1. Execute an nmap scan.
  2. Run a gobuster scan based on nmap results.

3. Execute an ffuf scan.

4. Run a sqlmap scan.

- **Output:**

The results of each task are logged and displayed in the Streamlit UI.

## Testing & Validation

- **Unit Tests:**

Unit tests are implemented using Pytest, covering:

- Task execution flow.
- Scope enforcement logic.
- Error handling and retry mechanisms.

- **Running Tests:**

bash

CopyEdit

pytest tests/

## Demo & Documentation

- **Video Demo:**

A video demonstration is available that shows the setup, execution, and outputs of the pipeline.

- **Further Documentation:**

Detailed explanations and configuration options are available in the project's Wiki.

## Contributing

Contributions are welcome! Please fork the repository, make your changes, and submit a pull request. For major changes, please open an issue first to discuss your ideas.

## **License**

This project is licensed under the MIT License.

## **Contact**

For further information or questions, please contact [Your Name] at [your.email@example.com].

yaml

CopyEdit

---

Simply copy the entire block above into a Word document, adjust formatting if needed, and save it as a .docx file.

o3-mini