STATE SPACE SLEUTH

Jesse Franks

INTRO

Overview of Problem and why it matters:

- Software testing is tricky, and it's hard to cover every combination of inputs a user can give. While creating and testing robust software helps to mitigate risk, you can never achieve 0 risk. Software with security vulnerabilities can leads to a wide range of problems, including loss of personal data, negative PR, and lost revenue.

Objective:

- Create an agent that can expose SQL injection vulnerabilities in an application

PROPOSED SOLUTION

Our agent can explore the state space of an application and intelligently work to find erroneous states in it. The approach will use A*, and a comparison to BFS, to do this. The state space will be represented by the response of our login page, where the algorithm aims to login as an Admin without providing a password. SQLite will be used to host the DB. A lightweight Flask app will be used as the “victim”, and BeautifulSoup will be used to help parse HTML responses.

Deliverable: An intelligent agent that iterates on predefined username and password variations to attempt to login to our application as an Admin without providing a password. The path the agent takes will be tracked along with other metrics that seem fitting as the project is developed. These metrics will be used to document the process and as a comparison between the DFS and BFS approach.

TECHNICAL DETAILS

Roadmap:

1. Create a lightweight app with a /login endpoint, where there are username and password fields
2. Create the DB to store our users in
3. Implement a State class to hold information on a state, such as the cost(s), username, password, and parent State. We will also implement a __le__ method to help compare states in the heap
4. Implement helpers to calculate costs and generate mutations
5. Implement the A* and BFS algorithms
6. Write a script to log information throughout the process.