SecureInT Visualizer Implementation & API (backend) documentation

21/07/2017

By Saket Khandelwal

NUS school of Computing

1. **Introduction**
   1. **Scope**

SecureInT is a discrete-event simulation model to investigate, analyze and secure insider threats. In order for users to use this model we need a middleware. Thus, a web visualizer was proposed to visualize and to utilize the properties of the simulation model in which users will be able to interact with the model through this interface.

The final scope is to design an interface such that a user can visualize the network, the users and the interactions. The user should be able to tweak parameters like vulnerabilities for the hosts and connections between users. Furthermore, the user should be able to "play" the simulator and view metrics like the vulnerability for the overall system. The frontend should be designed in such a way that it is easy to plug in any simulator (API for backend in Python).

* 1. **Overview**

The visualizer is developed as a web interface in which user can interact with the interface from anywhere. The interface is implemented using web programming tools and python framework as a server-side handler.

This project is implemented using HTML, CSS, JavaScript and Python. Flask is the python framework used in order to maintain and join the back-end with the front-end. Angular JavaScript is used to implement routing and scope to bind the controller with the view (HTML).

 

1. **High Level Description**
   1. **Architecture Diagram**

Front -End

HTML/CSS

* Rendering templates
* Views

Angular JS (Controls activity)

* Controller



Back- End

Flask (Python)

* API functions
* Routing
* Data communication
* Server initiator



* 1. **Architectural Diagram Description**
     1. **Front-End**

The views are implemented using HTML and CSS which render the templates. The controller is implemented using JS and Angular JS binds the view and the controller. As the user input is analyzed by the control and then views are manipulated by the controller. The front-end takes care of the visualization part as the network graph and simulation graph are created using D3 library (JavaScript).

* + 1. **Back-End**

The back-end or server-side is implemented using Python on the Flask framework. The back-end does the routing of the pages, creates API functions for external simulators and manipulates the data stored. The controller uses JS to trigger API functions which call the python functions in the back-end ergo flask is consumed by the controller.