A Visual Approach to Symbolic Execution

Nick Pfister - Astrophysics

Mentors: Fish Wang, Christophe Hauser, Yan Shoshitaishvili

Faculty Adviser: Christopher Kruegel
Department of Computer Science

Our safety depends on software!

What happens if this software fails?

We may analyze software using **Symbolic Execution** to...

- Examine how software works
- Detect vulnerabilities
- Detect malicious software - aka **malware**

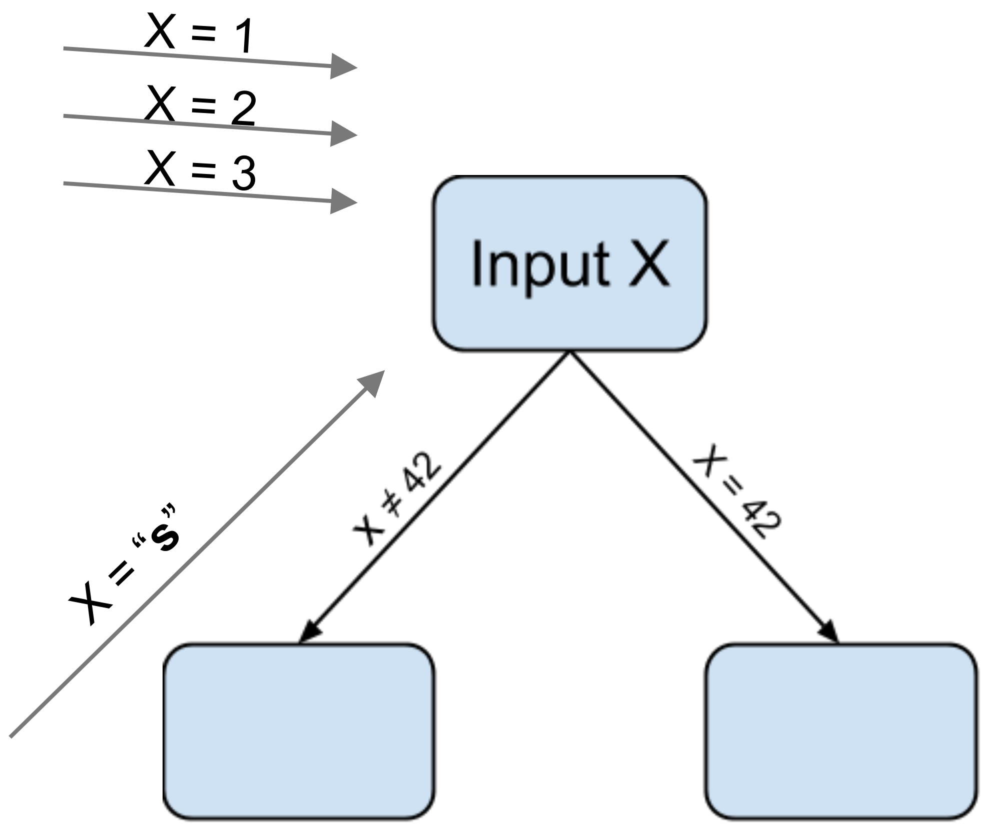
Why Symbolic Execution?

2 Types of Analysis

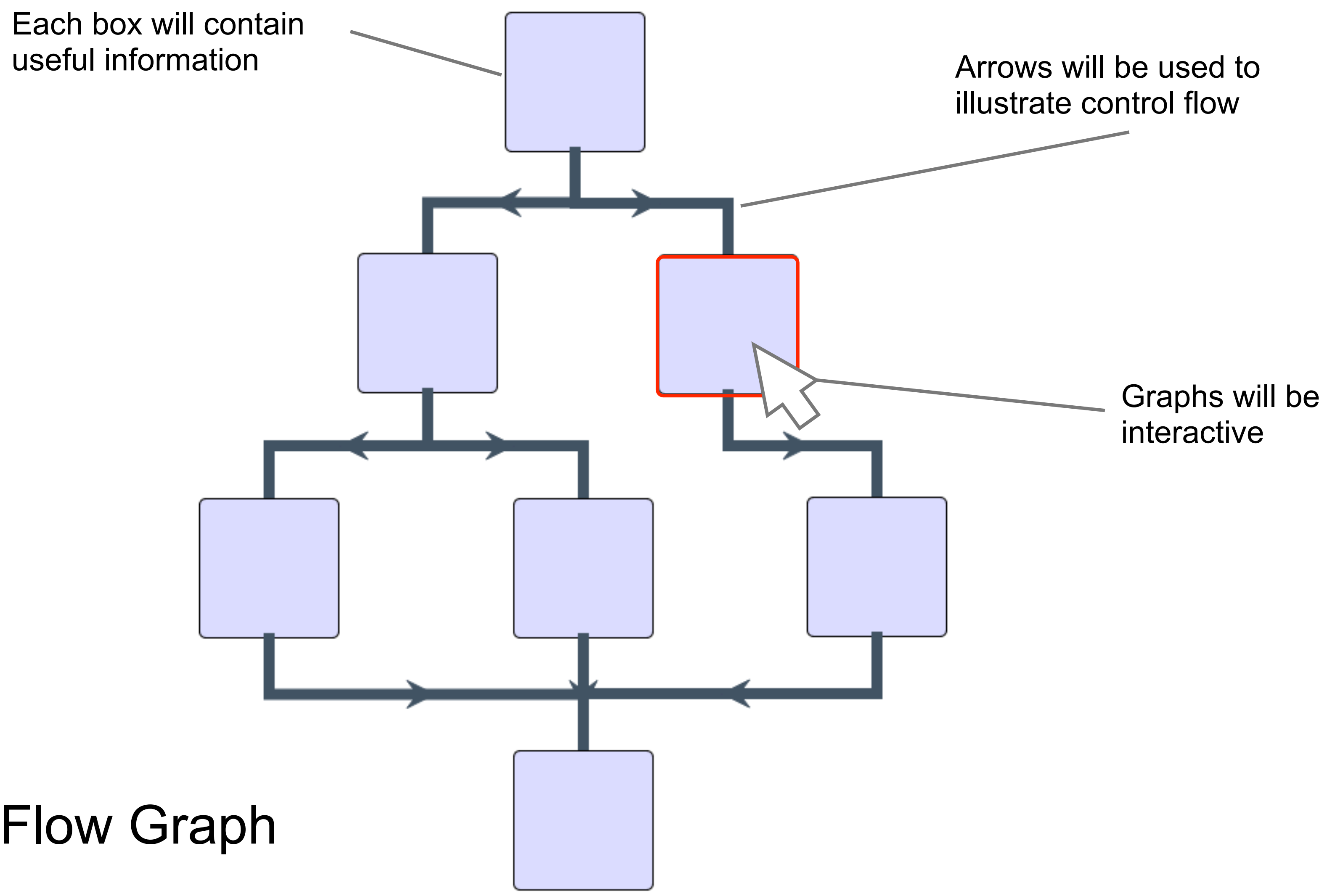
Dynamic Analysis
Runs program many times with many different inputs

Static Analysis
Examines the source code of a program, but doesn't execute it

Symbolic Execution, a type of Static Analysis, inputs an abstract variable and solves the value for all pathways

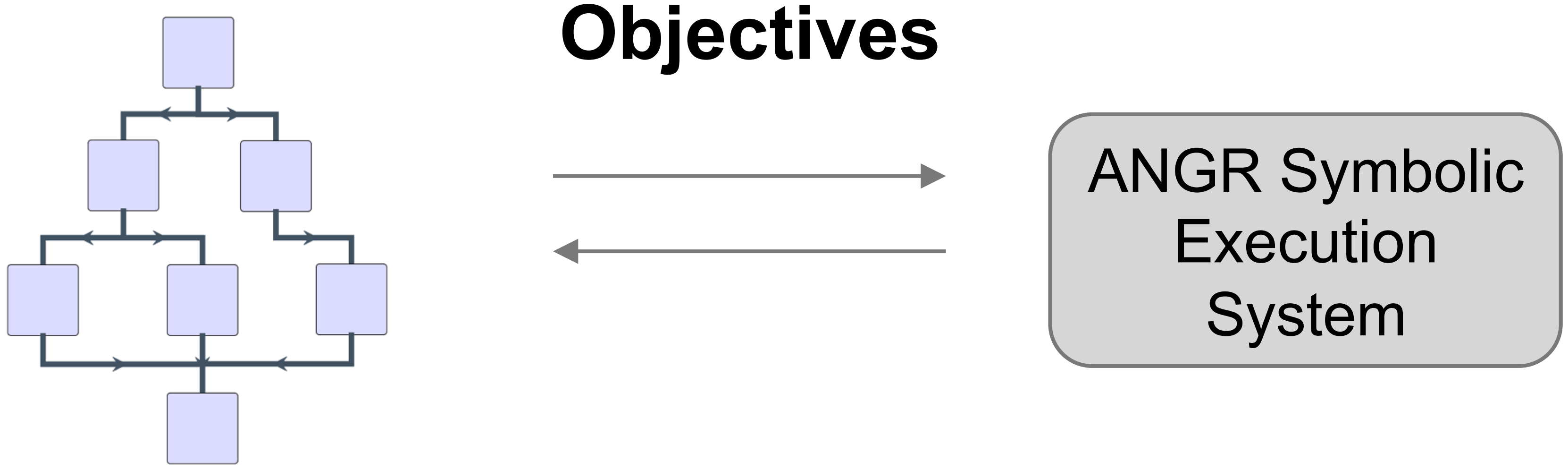


Visualizing Symbolic Execution


Each box will contain useful information
Arrows will be used to illustrate control flow
Graphs will be interactive

Control Flow Graph (CFG)

Objectives



Frontend
“Visual/Abstract end”

Backend
“Operational End”

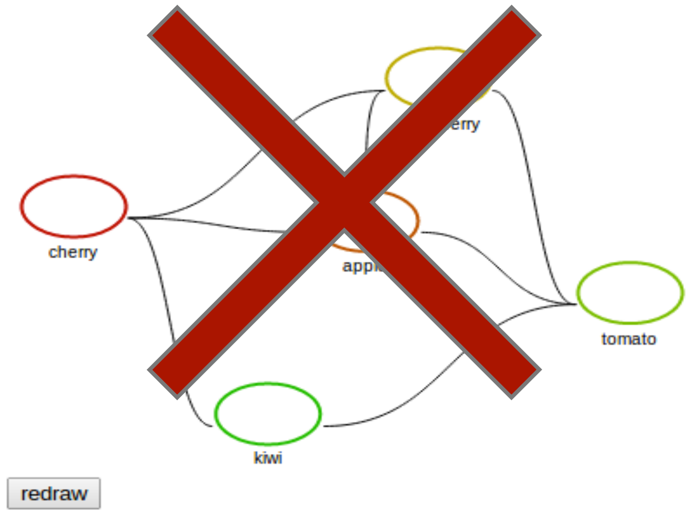
With no connection, these are not useful

With a little coding, we can create a connection!

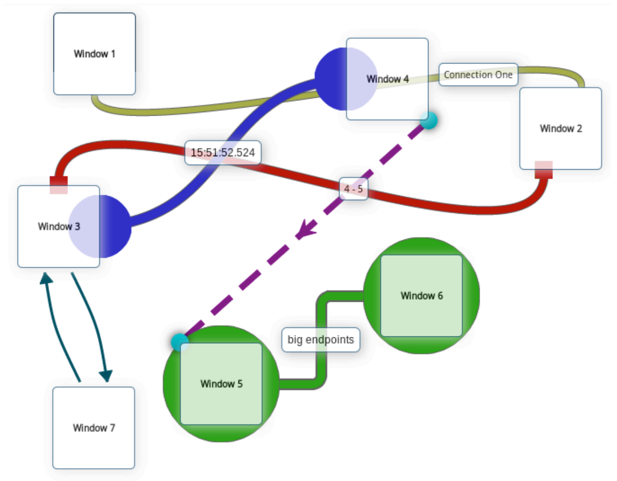
Identify and Evaluate Useful Libraries

Useful visual and backend libraries already exist

Determine what works best for our application

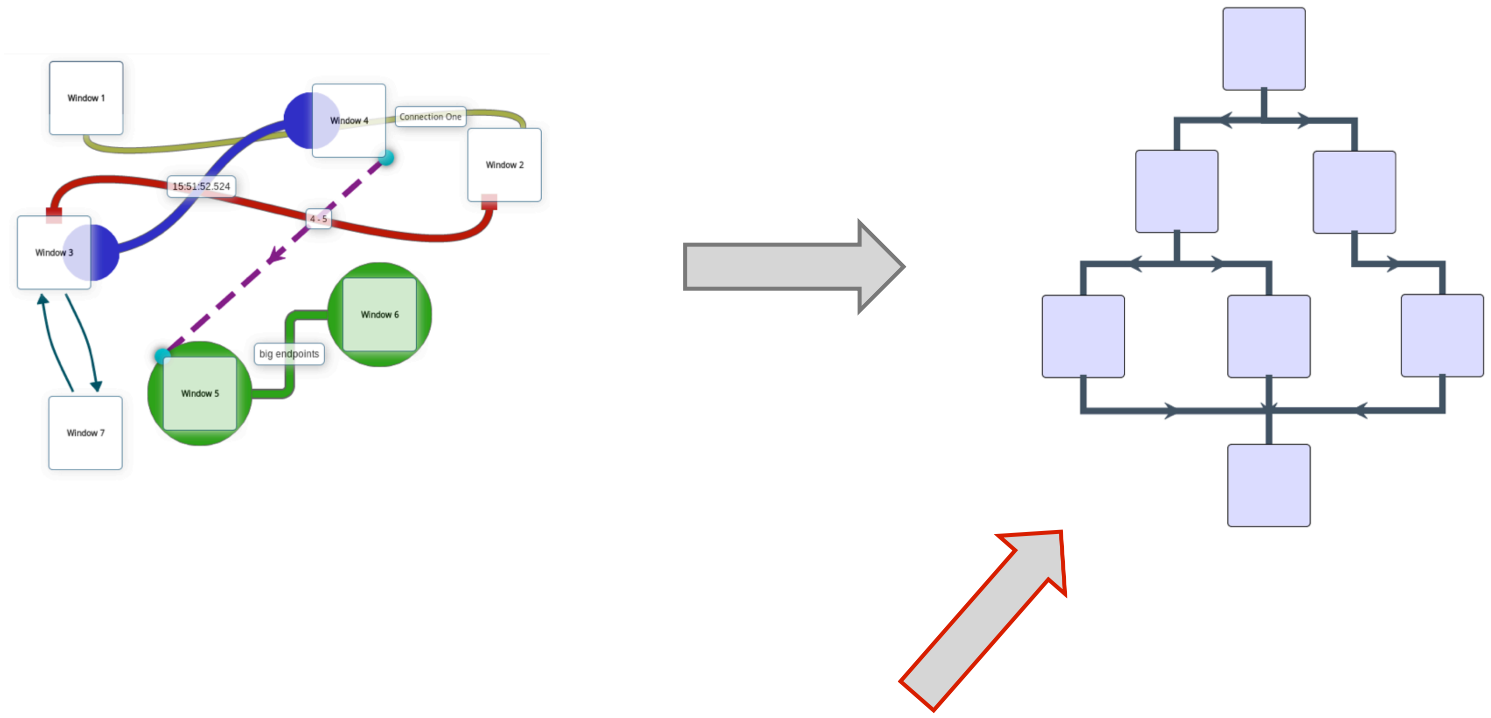


GraphDracula



JSPlumb

Implement Libraries



ANGR Symbolic Execution System

Backend Software

Experimental Data

We can measure the effectiveness of our visualizations by examining it’s speed and usability

Visual Application
Average Loading time (20 trials): 5.15ms
max: 11.03ms **min:** 4.09ms

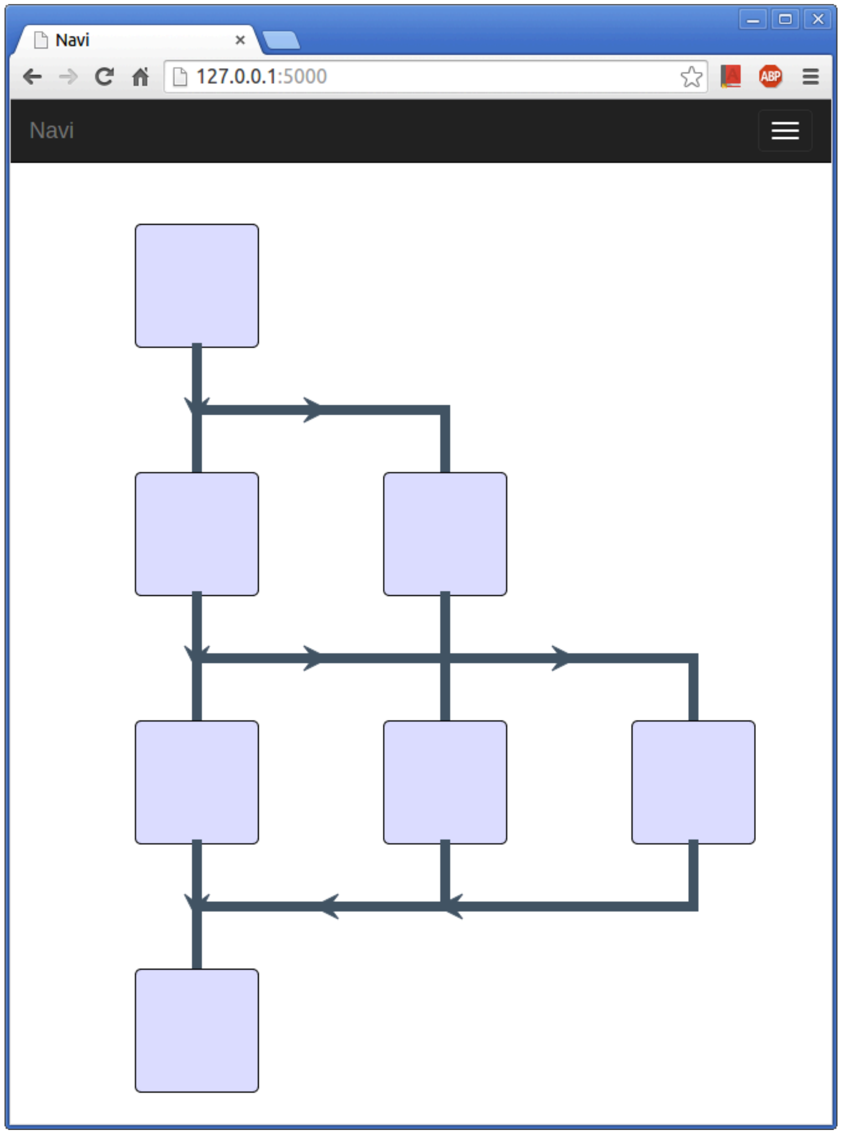
A 5ms loading time is negligible when compared to the backend processing time

As this interface improves to handle more complex graphs, loading time will have to be re-assessed

Interface

Snapshot of our web-based user interface

Our interface will be user-friendly and easy to understand

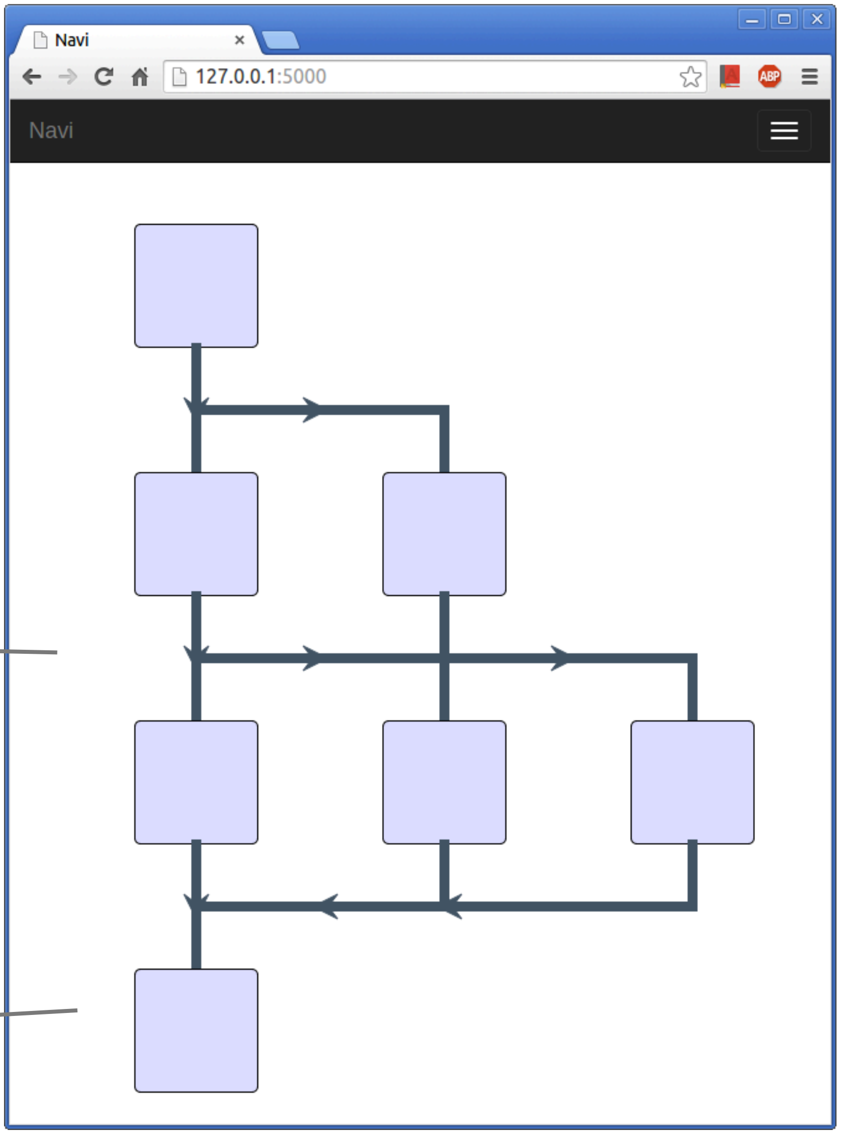


Interface

Preliminary Interface at startup (test without program data)

Connections are unclear

Containers are small/lacking detail

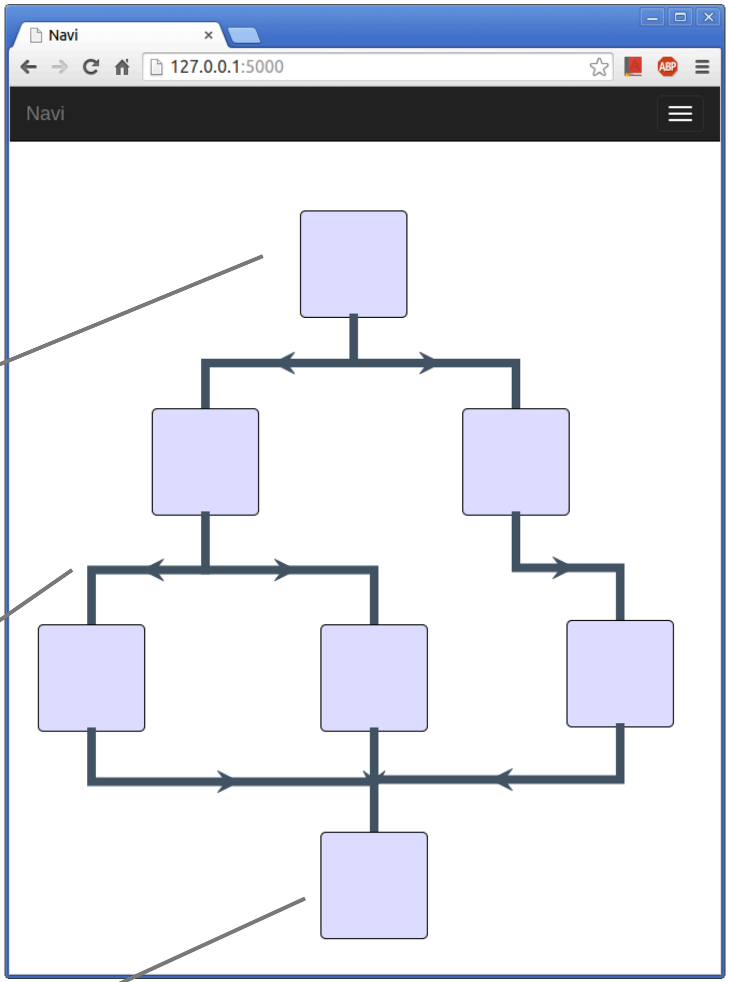


Interface

Interface after first revision (test without program data)

Containers can be reorganized by user

Connections are more visible



Containers are still small and not interactive

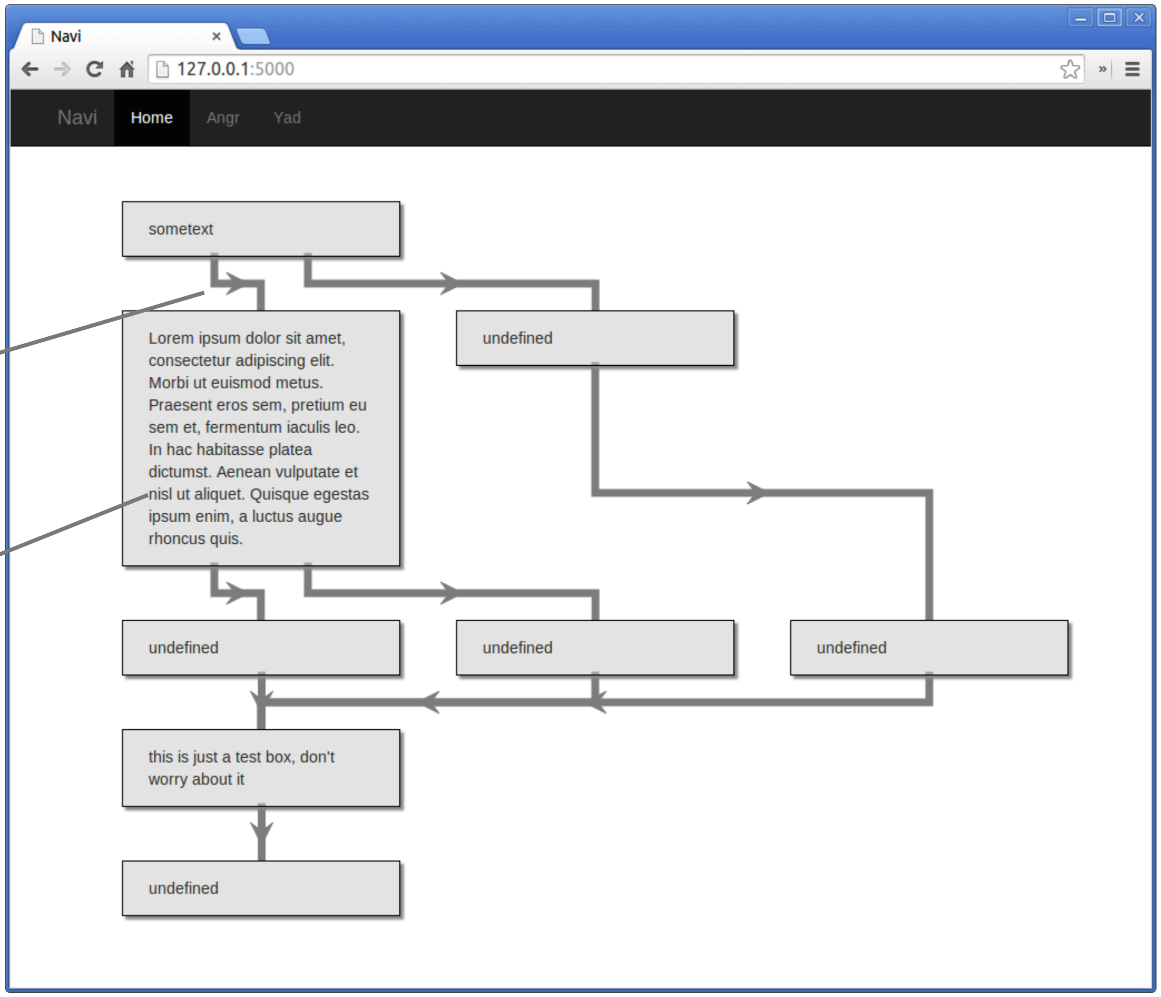
Interface

Interface after most recent revision

Improved connections

Resizable containers

Improved graph organization



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi ut euismod metus. Praesent eros sem, pretium eu sem et, fermentum iaculis leo. In hac habitasse platea dictumst. Aenean vulputate et nisl ut aliquet. Quisque egestas ipsum enim, a luctus augue rhoncus quis.

this is just a test box, don't worry about it

Future Plans

This interface is part of a much larger project, and will continue to be improved

The coding behind this interface is currently being implemented by researchers in the SecLab to visualize CFGs at DEFCON

Additional revisions to the interface are planned to make generate graphs of larger, more complex programs

Achievements

Previous computer skills: Java, C, C++

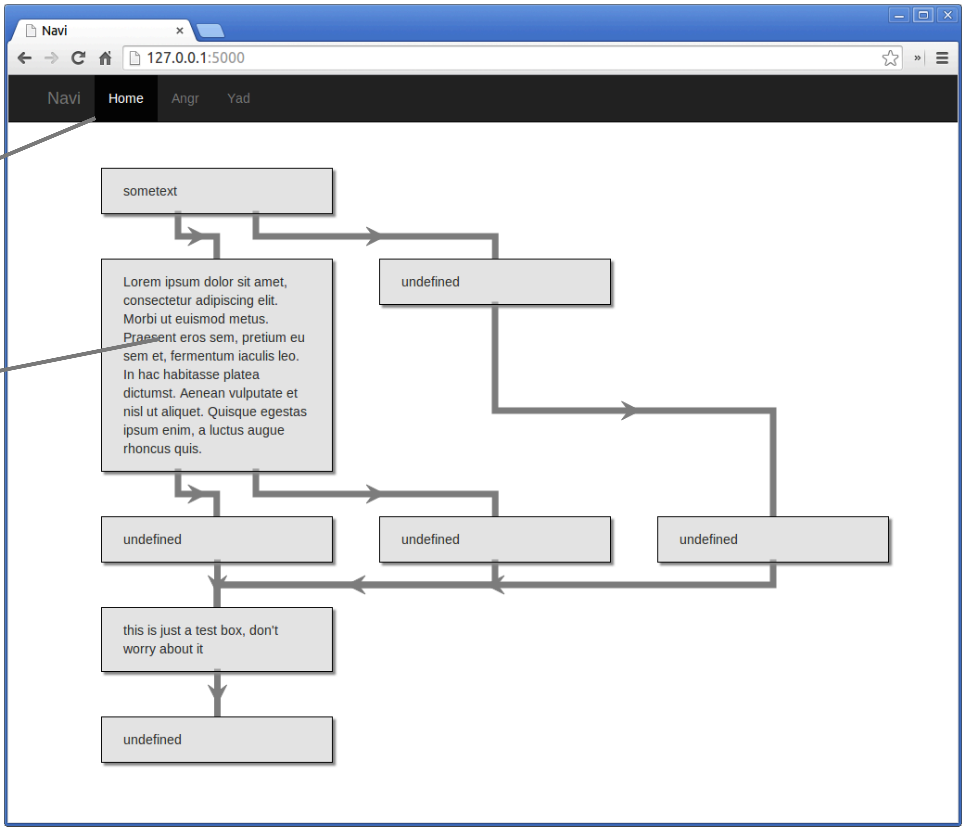
To create this interface, I had to learn...

HTML/CSS for website layout/styling

JavaScript/jQuery for graph generation

Python for backend development

Git/GitLab use for sharing and merging code



Navi
127.0.0.1:5000
Navi
Home
Angr
Yad
sometext
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi ut euismod metus. Praesent eros sem, pretium eu sem et, fermentum iaculis leo. In hac habitasse platea dictumst. Aenean vulputate et nisl ut aliquet. Quisque egestas ipsum enim, a luctus augue rhoncus quis.
undefined
undefined
undefined
undefined
this is just a test box, don't worry about it
undefined

Achievements Continued...

Most importantly,developing this software has given me first-hand experience with...

Organization/planning
Experimental methods
Trial and error

“I have not failed. I have just found 10,000 ways that won’t work.” -Thomas Edison

Acknowledgements

Special thanks to...

Mentors
Fish Wang, Christophe Hauser, and Yan Shoshitaishvili

Faculty Advisor
Christopher Kruegel

INSET
Maria Napoli, Jens-Uwe Kuhn, Nick Arnold, my fellow interns, and everyone else involved

In addition,
Todd Brei, Mike Young, Stephen Strenn, and Jerry Wyss for nurturing my interest in math and science