# Visualization for Secure Coding in C

*James Walker, Jean Mayo, Ching-Kuang Shene (Michigan Tech)*
*Steve Carr (Western Michigan University)*

## Self paced

Visualization is driven by events generated through static and dynamic program analysis. Students control the pace at which the events are played. They may also step, rewind, and fast forward through the event queue.

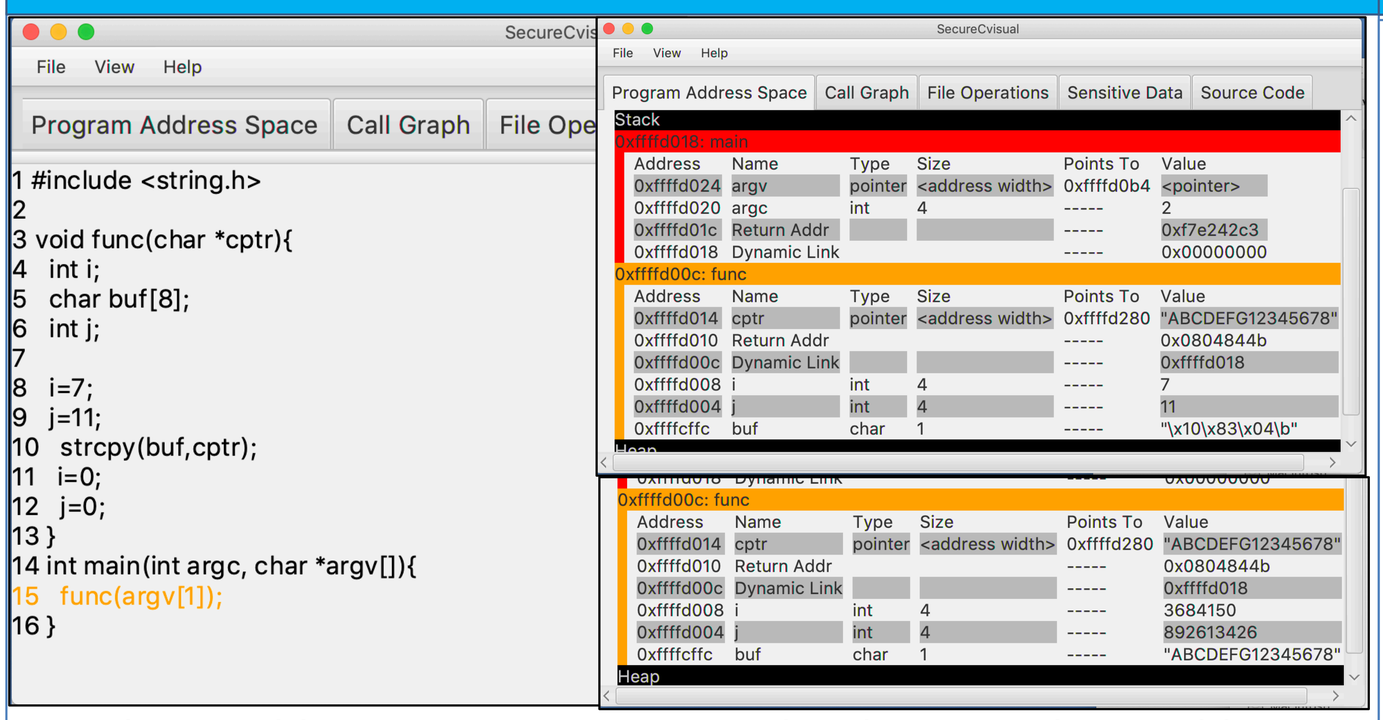
## Multiple Learning Levels

Four experience levels determine the set of visualizations shown by default and the details that appear in the Program Address Space visualization. Designed for beginning C programmers through graduate students.

## More than Security

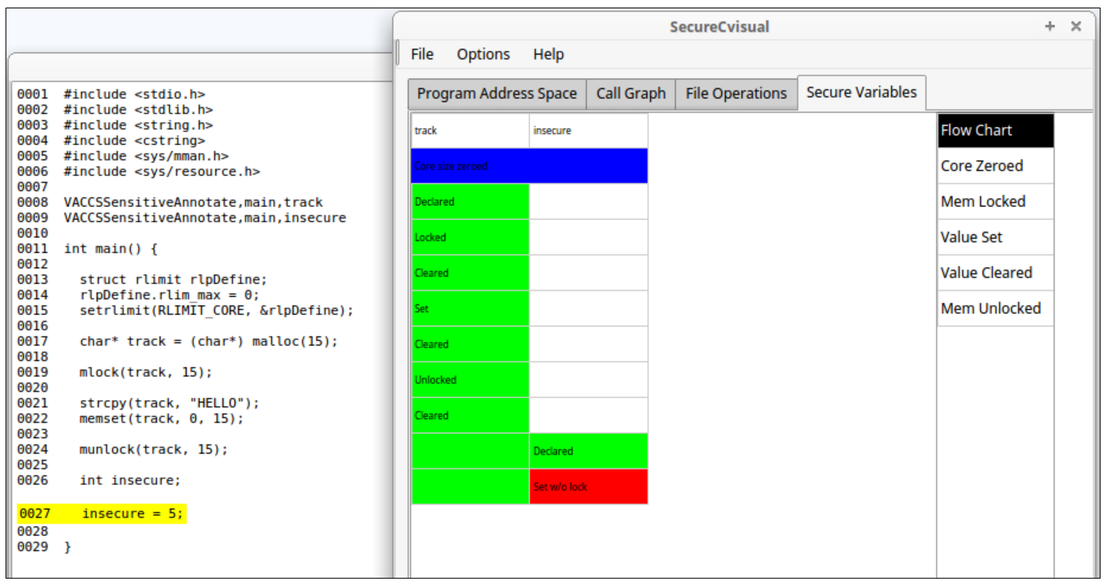
Visualization of the program address space, call graph, and basic kernel file handling structures can be useful in many courses, including C programming, Systems Programming, and Operating Systems.

## Program Address Space



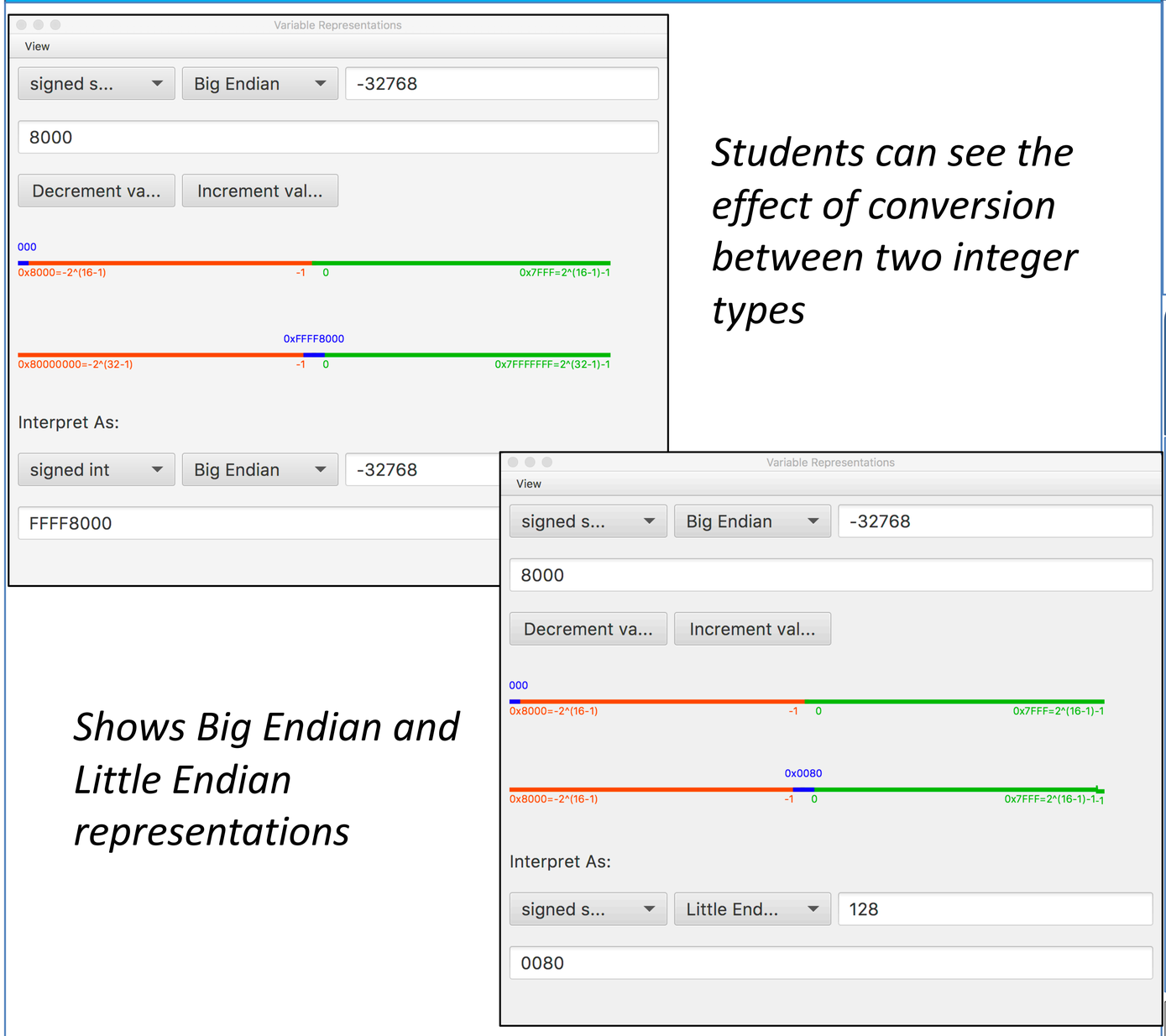

- Shows address space, registers, and code in C and assembly.
- Choose among multiple levels of detail.
- Easily demonstrate heap and stack overflows

## Sensitive Data Handling




- Students formally identify sensitive variables.
- Visualization identifies if data properly protected: locked in memory & core size set to zero before use; data cleared before unlocked

## Integer Coercion




*Students can see the effect of conversion between two integer types*

*Shows Big Endian and Little Endian representations*

- Shows twos complement representation of any integer variable
- Students can see effect of conversion between two integer types
- Depicts equation evaluation and identifies coercions that will be performed

## File Operations*

- Shows students basic kernel file handling data. Works across fork. Shows per-process and global structures.
- Each open depicted and effect of parameters explained.
- Helps students understand shared descriptors and to apply principle of least privilege in file operations.

## Future Work

- Eclipse plug-in under development. Planned features include:
  - Static and dynamic analysis
  - Real-time syntax highlighting
    - Attack surface, malloc, free, open, etc.
  - Tutorials on fundamental concepts like attack surface and buffer overflows, etc.
- Planned support for dynamic taint analysis
- Contact: Jean Mayo (jmayo@mtu.edu) or Steve Carr (steve.carr@wmich.edu)

This material is based upon work supported by the National Science Foundation under Grants DGE-1523017 and DGE 1522883 with additional support from DUE-1245310. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

* Under development