



# *How to Break Software*

James A. Whittaker, Ph.D.


Professor of Computer Science

Director of the Center for Information Assurance

Florida Tech



## *What You Will See in This Talk*

- Lots of bug demos
    - Live demos of really cool bugs you probably haven't seen
    - New security bugs you surely haven't seen
  - Lots of ideas to help you find bugs faster
  - Cool new testing tools that will allow you to reach functionality you've never been able to test before
- 



## Users Use, Testers Test

- Any fool can stumble across bugs
  - Flailing around hoping to find bugs is a job for amateurs
  - Testers celebrate bugs!
- Real **testing** requires:
  - *efficiency*: finding bugs faster than normal use
  - *effectiveness*: finding bugs that users care about and that developers will fix
  - *thoroughness*: leaving no stone unturned



## Mastering Software Testing

- Think about testing as an:
  - Art
  - Craft
  - Discipline
- Testing is a **discipline**; to master it, one must train properly
  - Strive for individual excellence
  - Ensure group learning takes place



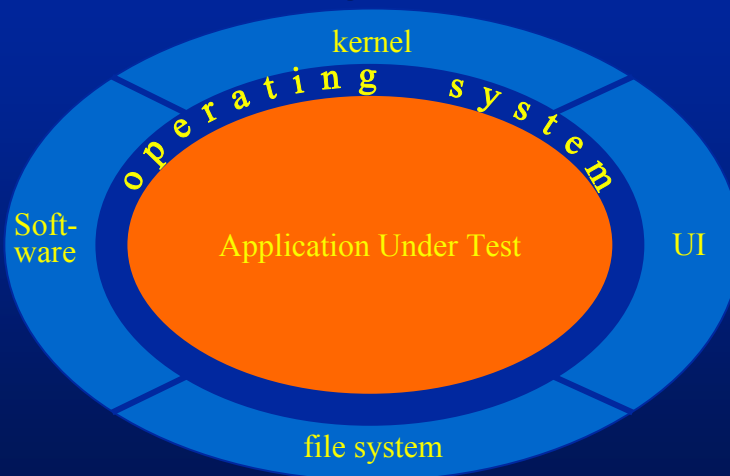


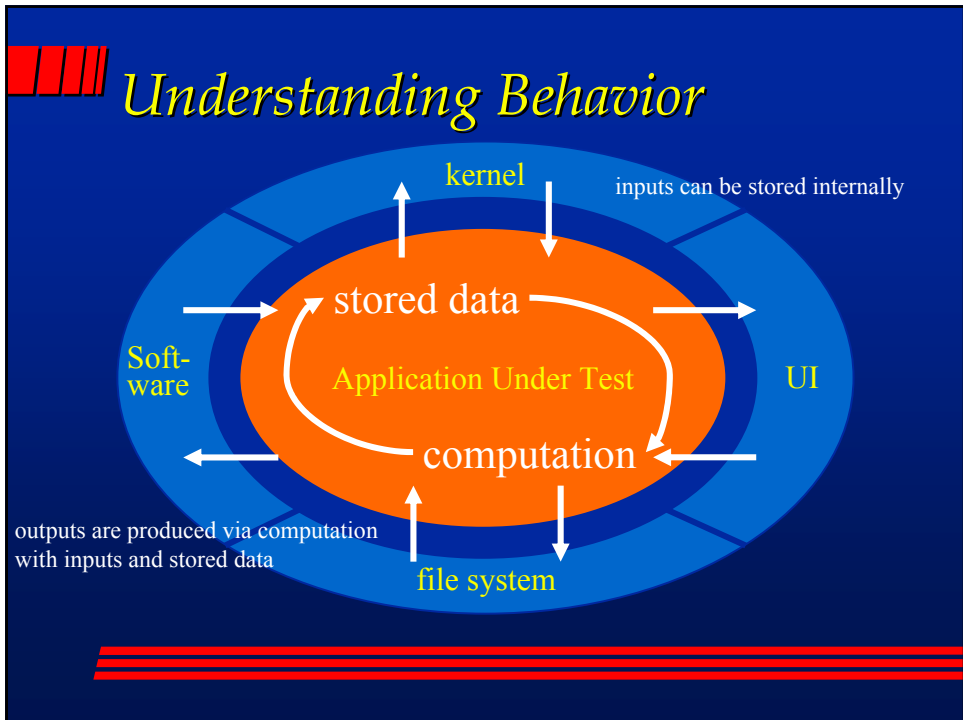
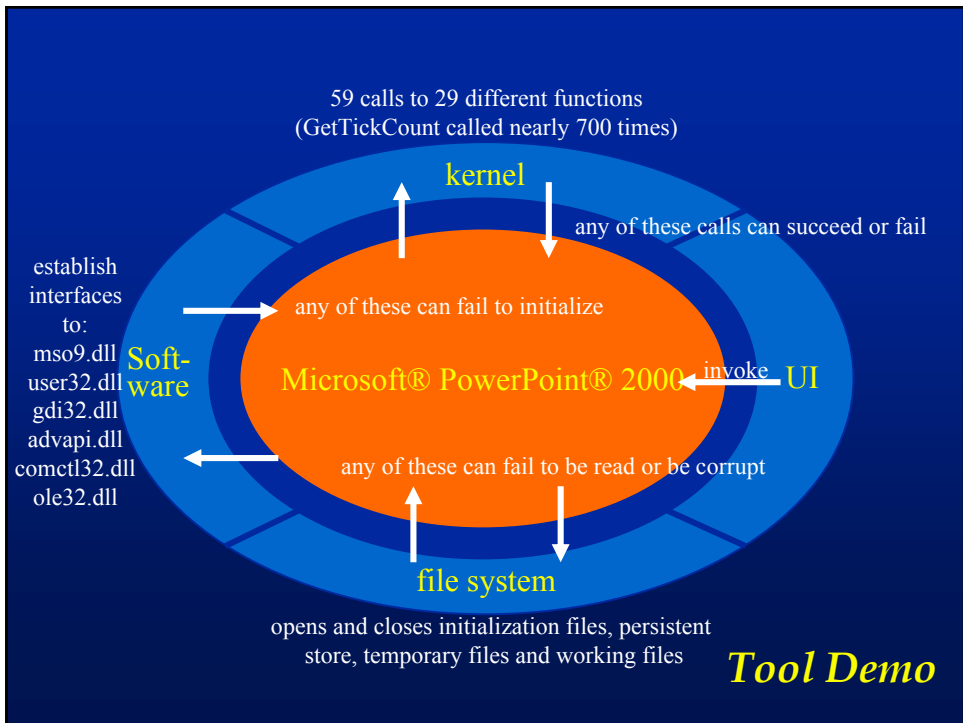
## Software Test Training

- Mastery requires the development of:
  - Understanding of:
    - » software
    - » bugs
    - » failures
    - » test techniques
  - Intuition concerning:
    - » correct software behavior
    - » defective software behavior
    - » and **patterns** of both



## Understanding Behavior





## *Testing Input Behaviors: Patterns of Attack*

1. Force all error messages to occur
2. Force the software to establish default values
3. Explore allowable character sets and data types
4. Overflow input buffers
5. Find inputs that interact with other inputs
6. Repeatedly apply the same input/input sequence

 *Bug Demo*

## *Testing Output Behaviors: Patterns of Attack*

7. Force different outputs to be generated for each input
8. Force invalid outputs to be generated
9. Force output properties to change
10. Force the screen to be refreshed

 *Bug Demo*

## *Testing Data Behaviors: Patterns of Attack*

11. Apply inputs using a variety of initial conditions
12. Force a data structure to store too many/too few values
13. Investigate alternate ways to modify internal data constraints

 *Bug Demo*

## *Testing Computation Behaviors: Patterns of Attack*

14. Experiment with invalid operand and operator combinations
15. Exploit recursion
16. Force computation results to be too large or too small
17. Find features that share data or interact poorly

 *Bug Demo*

## *How To Contact James A. Whittaker*

### **Snail Mail**

Department of Computer Sciences  
150 W. University Boulevard  
Melbourne, Florida 32901-6975

### **E-mail & Web**

[jw@se.fit.edu](mailto:jw@se.fit.edu)  
[www.howtobreaksoftware.com](http://www.howtobreaksoftware.com)

### **Telephone**

321 674 - 7638

### **Fax**

321 674 - 7046

# *The End*

Questions?

Comments?

Bug Stories?



## *References*

- J. A. Whittaker, *How to Break Software* (Addison Wesley, Reading MA, 2002).
  - J. A. Whittaker, "Software's invisible users," *IEEE Software*, 18, 3, pp. 84-88 (2001).
  - J. A. Whittaker and H. H. Thompson, *How to Break Software Security* (Addison Wesley 2003).
- 