



## INCREASING SECURITY IN AVIATION-ORIENTED COMPUTING EDUCATION: A MODULAR APPROACH

[Project Info](#)[Modules](#)[Related Links](#)[Papers](#)[Project Team](#)

# Instructors' Guide for Demonstrating Buffer Overflows

Below are some tips on how to effectively use the applets associated with the [Buffer Overflow Vulnerabilities module](#) of NSF Grant ["Increasing Security in Aviation-oriented Computing Education: A Modular Approach"](#).

## Demonstrating the applets - Mechanics

- Projection and Computer Setup
  - For a classroom, you'll need a standard overhead projector and a computer to step through the applets. Important: make sure you have a mouse or other device you're comfortable with.
  - For labs where students access their own computers, you may need to adapt these instructions
- Media
  - If you're on a fast and reliable network, you can run directly from the web
  - Alternatively, the website contains three compressed packages you can decompress onto a local hard disk or floppy, zip, or CD:
    - [Zipped](#)
    - [Tar](#) and [Tar.gz](#)
- Explanations of [parts of the screens](#) describes the layout of the applet window and buttons

## Preparation - Content

- Allocate at least 30 minutes for in-class presentations
- Make sure students know the concepts of run-time environments. Two applets provide this general overview
  - [Jumps](#): An introduction to the way languages like C use stack frames to store local variables, pass variables from function to function by value and by reference, and also return control to the calling subroutine when the called subroutine exits. This uses pseudocode in the place of C.
  - [Stacks](#): The same as above but using C instead of pseudocode.
  - For a background spiel, work through the [IntroToBufferOverflow](#) slides or in the [Authorware Interactive Learning](#) tool
- The applets will show how a buffer overflow occurs but not the various consequences which are outside the scope of the module"
  - Privileges, ROOT, etc. in operating systems
  - Denial of service in networks and security
  - Unauthorized access and disclosure of information, e.g. posting credit card numbers on the web
  - Modification or corruption of databases
  - etc.
- Remember the applets use a "virtual machine" abstracted
  - a small memory size, but big enough to show the problem
  - abstracted pointers into a stack stored in the memory
  - linking C code with resulting "compiled" code

## Doing the Demo

- These applets were written by a college senior to appeal to college students, hence the Spock theme.
- Two main applets capture the essence of the problem.
  - [Spock](#): Demonstrates what is commonly called a "variable attack" buffer overflow, where the target is data.
    - Play or step forward until data is requested (to be typed into the top box) - then enter an incorrect answer and run to completion, "access denied"
    - Run the applet a 2nd time and enter the correct password
    - Now invite the students to figure out what password data to type to gain access as Dr. Bones when they don't know the correct password and run the applet a 3rd time.
  - [Smasher](#): Demonstrates a "stack attack," more commonly referred to as "stack smashing."
    - This program has some "bad code", a subroutine that we hypothesize has some bad code in it, e. g. to hijack the computer as in our [Mailroom Metaphor story](#)
    - Run the applet and enter some normal data, showing the start and end of the data for each subroutine, including the return pointers
    - Invite the students to figure out how to "smash the stack", forcing control to go to "DontCallThisFunction". Hint: figure out the ascii character for the first address of that subroutine on the stack.
  - Defenses are also illustrated - you should read the referenced papers for background in explaining these
    - [StackGuard](#): This demo shows how the StackGuard compiler can help prevent "stack attacks."
    - [its4demo](#): Shows the output of ITS4, a static analyzer, on two different C programs.
  - Finally, here's a real live script
    - [bodemo](#): This is a mock attack on a linux system demonstrating how an attacker can get a root shell.

## Follow Through

You might now reinforce the following concepts:

- The ethics of hacking - responsibility for consequences, effects on IT time spent patching software. Plus any script kiddie can do this, it's no longer a challenge to attack a system - defense and counter-measures is where the professionals are.
- Checking input data for overflows, now you understand the possible effects
- Using different string functions
- Having a [checklist to use during code inspections](#)
- [The scavenger hunt](#) to find out more about the economics of buffer overflows.

---

[Project Info](#) • [Modules](#) • [Links](#) • [Papers](#) • [Team](#) • [NSF](#)

Last update: July 25, 2002