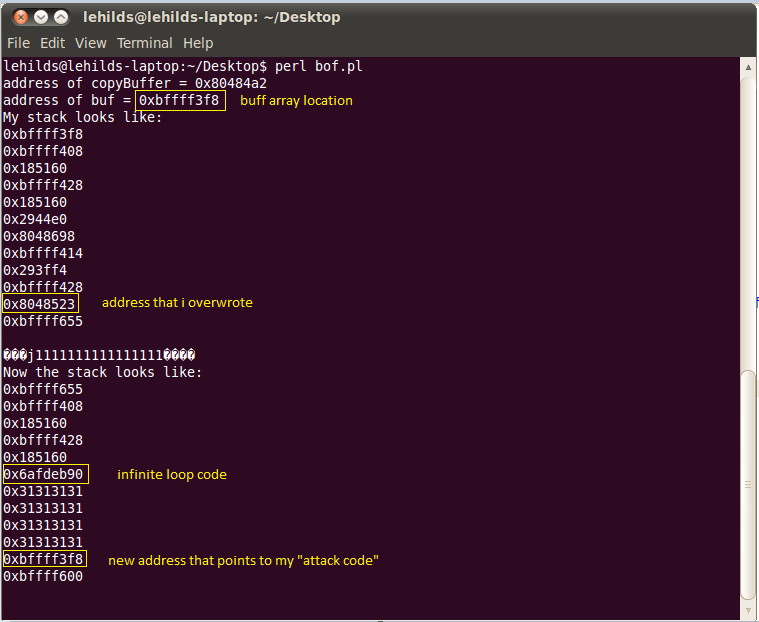
IT 252 HW 8 code

Name: Lehi Alcantara

Lines changed in perl file:

$arg="'\x90''\xeb''\xfd''\x6A'1111111111111111'\xf8''\xf3''\xff''\xbf'"; # input that will overide the return address of copyBuffer method

Screenshot of infinite loop:



**Explanation of what GCC options "-fno-stack-protector" and "-z execstack" do:**

1. **"-fno-stack-protector":**

Disables the stack-smashing protector.

The stack-smashing protector is: “Applications written in C will be protected by the method that automatically inserts protection code into an application at compilation time. The protection is realized by buffer overflow detection and the variable reordering feature to avoid the corruption of pointers. The basic idea of buffer overflow detection comes from [StackGuard](http://www.immunix.org/) system”. (<http://www.trl.ibm.com/projects/security/ssp/>)

1. **"-z execstack":**

Makes it so that the stack is executable, meaning you can execute a command from the stack.

**Explanation on what Linux command "sysctl -w kernel.randomize\_va\_space=0" does, and what happens without it:**

1. **“sysctl -w kernel.randomize\_va\_space=0”:**

It makes it so that the stack location addresses are not randomized, therefore stack locations will be procedural, meaning an a hacker could use close locations to attack a code.

1. **what happens without it?**

Without it, it makes the stack location addresses randomized, meaning attacker will have difficulties trying to attack a code because stack locations will no longer be procedural. Therefore making it harder for a hacker to break a code. In our case, I get a segmentation fault, meaning: “when the hardware notifies a Unix-like Operating system about a memory access violation” (<http://en.wikipedia.org/wiki/Segmentation_fault>)