Buffer Overflows

Description

You should use your ubuntu instance to complete the following.

We are going to take a look at how to overflow a simple buffer of some c code to gain terminal access. Answer the following questions where indicated.

Here is a simple video to get you started.

In your home directory you should install two c programs, buffer.c and hack.c. These are found here and here.

Compile buffer.c gcc -o buffer buffer.c

Run it and type in a bunch of characters and hit enter. To run it, you just type ./buffer. You may have to chmod +x to make it executable. It then waits for you to type in a string. Type one in and see what happens.

Type a very long string in, you should see something like *** stack smashing detected ***: ./buffer terminated.

Make a note of the maximum number of characters that you can type in without getting the above error?

Now, recompile the above code without stack smashing protection

gcc -fno-stack-protector -U_FORTIFY_SOURCE -o buffer buffer.c

Run the code again with lots of characters.

What does the new error message say?

Run the code 3 or 4 times

Record the address of where that is trying to run each time. The address is indicated by the value of buffer, something like buffer = 0xbffdd09c0.

Note that this address changes each time you run the program. Why does that change?

Disable address space randomization in linux by doing: sudo /bin/sh -c "echo 0 > /proc/sys/kernel/randomize_va_space". (If you need to re-enable it, you can change the 0 to a 1)

Run your code again 3 or 4 more times.

What happens to the address now.

Why does it not change?

Compile hack.c gcc -o hack hack.c

We are going to feed the output of hack.c (which generates some specially crafted input) into our buffer program.

First we will recompile buffer.c one more time gcc -fno-stack-protector -z execstack -o buffer buffer.c

What do the options fno-stack-protector and execstack do? (See google)(I will ask you this on your submission file)

You will have to do a apt-get install execstack

Verify that the execstack is appropriately set by issuing execstack -q buffer (just make sure there aren’t any weird errors)

Now issue the following:

./hack [buffer address] [diff] | ./buffer, where the inputs to buffer address and diff are given by a run of ./buffer

Ideally now you have a shell, try to type ls and hit enter. (Ctrl-D to exit the shell)

Take a print screen of your buffer overflow.

To submit

A single pdf with the answers to the following questions. Many of these answers will require you to do some research on your part.

What is a buffer overflow?

How does address space randomization mitigate buffer overflows?

How else can you prevent buffer overflows?

What do the options fno-stack-protector and execstack do? (See google)

Find a recent vulnerability of a buffer overflow and report what program it affects and anything else interesting about it.

Include a screenshot of your above buffer overflow working.