

Forum:[Hack Forums](#)

Thread: [\[Tutorial\]Buffer Overflow Part 1: Introduction to Stack](#)

7 **R3X** - August 8, 2017, 1:13 pm

Stack Buffer Overflow

I've been learning about Buffer Overflow for quite sometime and I thought it'd be nice to share my learnings with others and this is the reason for this Tutorial. This is the first part of the tutorial with more coming later.

Who is this Tutorial for?

This tutorial is for beginners who have little programming experience and wants to know what BOF is and how exactly does it work.

What will we learn?

We'll look at a very simple demonstration of how we can alter the flow of a program with correct knowlegde of the allignment of stack and a precisely formed malicious input for a vulnerable function with a program.

Prerequisite:

Let's prepare our guns before we dive into a battle.

Stack

A geeky [definition of stack from Wikipedia](#) says,

Quote:"In computer science, a stack is an abstract data type that serves as a collection of elements, with two principal operations: push, which adds an element to the collection, and pop, which removes the most recently added element that was not yet removed. The order in which elements come off a stack gives rise to its alternative name, LIFO (for last in, first out)."

Basically, all stack does is to store elements and retrieve them in a very certain order. Only the most recent element inserted (or pushed) on to a stack can be retrieved (or popped). The position of this most recently pushed element is also called the "top" of the stack. The working of a simple stack is show in the diagram below.

Consider we're working in a 64-bit environment i.e. each of our address is a 8 byte address. Also consider that the stack starts at a memory address of 0x1000("0x" means hexadecimal). The image shows how 64-bit integer would be pushed on to the stack and how the location of "top" changes as the stack grows. The memory address painted in red represents the "top" of the stack before the push operation is carried out.

Figure 1 shows only one element (integer value 1) resides at the top of the stack at memory address 0x1000. Since it is the only element in the stack, it is also the top of the stack. As we're using 64-bit integers, it'd take 64 bits or 8 bytes to store it. So we have memory locations 0x1000 to 0x1007 holding our first integer. In figure 2, second element (integer value 2) has been pushed to stack at memory location 0x1008, also making it the top of the stack. This continues till figure 5 when we have memory location 0x1020 as the top of the stack holding an integer value of 5.

Note: As I've already mentioned, the state of "top" in these figures is marked before the push operation is conducted. So, in figure 1 0x1000 is the top before the second element is pushed.

The counter part of push operation can be seen in the image below. It's called the "Pop" operation. It deletes the last top element from the stack and decreased the value of "top" to point to the next element in the stack.

A program always maintains a stack for its functioning. If you're a coder, you'd must have heard at least once that a stack is needed for making function calls and that's exactly what we're going to in the next part of this tutorial.

It's taking longer than I expected, especially because I'm new to GIMP (Yes, GIMP and not Photoshop. Why? Because I love Open-Source.). I'll continue to write this tutorial on Buffer Overflow as soon as I get time. If you have any questions so far, please leave them below in the comments and I'll try to answer them all. I encourage you to put some comments, either good or bad. Let me know if you liked the way I write, it'll encourage me to write further. If you didn't like the way I write, let me know about it too. One learns the best from his own mistakes. Your valuable comments are heartily welcome.

Edit: : I have written the sequel to this tutorial. Go check out [\[Tutorial\]Buffer Overflow Part 2 : Introduction to assembly](#)

You can buy me a cup of coffee so I could stay up at night and write for you.
My Blockchain wallet : 17mKvXY27BdS2FnPAs91FbvX2XuGos4BNW
Let me know if you've donated.
Have a great day ahead. Keep Hacking ?

andmuchmore - August 8, 2017, 3:14 pm

Nice tutorial! BOF's are always fun to hunt and exploit! Learning the fundamentals of exploiting a BOF is a must imo!

7_R3X - August 9, 2017, 2:48 am

(Aug 8, 2017 03:14 PM)andmuchmore Wrote: [\[CITE\]](#) Nice tutorial! BOF's are always fun to hunt and exploit! Learning the fundamentals of exploiting a BOF is a must imo!

Thanks for the comment. I really appreciate it. I'm working on the next part of this tutorial and put it online hopefully in a few hours. Cheers!