Chritopher LaJon Morgan
Oct 28, 2013

Homework 10

CS 465

Answer the Following:

- **What are two different ways to succeed at a stack smashing attack described in the paper?**

The two different ways to perform a stack smashing attack as described in the paper are:
1) Using a NOP sled
2) Using an environment variable

A NOP sled style of attack places NOP instructions and some exploit code directly in the buffer, or more specifically into some user input being read into a buffer. The goal then is to override the return address to point into the NOP sled, which will then skid the processor directly into the exploit code.

The other style of exploit follows a similar pattern but places the exploit code in an environment variable. This is desirable when the buffer one is overflowing is small, the shell code will not fit in the buffer/input string size available, or the number of NOPs one can pad the front of the buffer with is too small to merit any help. The environment variable therefore is a good place, given that it can hold an arbitrary amount of exploit code and that its address is set at the top of the stack when the program is started. Thus, the exploit becomes overwriting the return address to point to the environment variable in which one has placed exploit code.

- **How does the paper recommend you find a buffer overflow vulnerability? Do you know of any other ways to find this vulnerability?**

The predominant assumption of this paper's recommendations on finding a buffer overflow vulnerability is that one can find out something about the source code. They're recommendations are all derived from the fact one can find something descriptive about the source code. Therefore, they recommend looking for library functions that do not perform bounds checking: strcat, strcpy, sprinf, vsprintf getc, fgetc, getchar, and grep. All these methods, in the right context, can provide the opportunity for a buffer overflow attack. One way that they recommend gaining access to the source code is by looking at free software utilities and operating systems. Their insight is that most commercial software is based on the free ones, therefore, use the free ones to guess about the commercial ones.

A required tell tail sign of a buffer overflow vulnerability is a program that uses some sort of user provided data. One adhoc way to check for a vulnerability is to find a program that takes user input of some sort and simple throw a massive amount of it at the program. Then simply check to see how the program reacts, if it crashes or gets into a bad state there is a good chance that an overflow vulnerability exists.