Anthony Constantino
HW 10

Stack Smashing

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

The first method causing a buffer overflow which then can allow them to overwrite the return address allowing them to open a shell. This then allows them to run whatever they choose.

Second there is a small buffer overflow. In this case, they overflow the buffer but it is too small for them to modify the return address to open the shell. Instead they can point to an environment, which they can overflow to open up a shell.

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

The paper suggests that many of the built-in C libraries used to read in buffers do not have very thorough boundary checking. In this case, you can make sure that you implement the boundary checking yourself to ensure it will not overflow.

Something that comes to mind is the types of loops that are implemented in the program. For example, a for loop has a set number of times that it iterates, but since the message might not always be that of a fixed length while loops might be used. This can be risky because they will go until conditions are met and leave themselves vulnerable to overflow if used to read in the buffer. Check while loops to make sure they cannot be used to perform this type of attack.