Simple Buffer Overflow Example

Dan Fleck

Reference: http://www.thegeekstuff.com/2013/06/buffer-overflow/
Buffer Overflows

- Buffer overflows occur when some sized portion of memory is overwritten with something bigger.

- For example:
  - char buff[10]; // Min index is 0, max index is ????

- Overflow:
  - buff[10] = 'd';
Why care?

• Minimally buffer overflows can cause program crashes or unexpected results

• char *ptr = (char *) malloc (10); // Allocate 10 bytes on heap
  ptr[10] = ‘d’; // Crash!

• However because buffer overflows corrupt memory nearby the variable being over-filled, an informed hacker can use this to write data into other protected areas of the program (i.e. stack and heap)

• char buff[10] = {0};
  strcpy(buff, “Lets overflow that buffer!”); // overflow!

Coming up: An attack
int main(void) {

    // Use a struct to force local variable memory ordering
    struct {
        char buff[16];
        char pass;
    } localinfo;

    localinfo.pass = 0;

    printf("\nEnter the password : \n");
    gets(localinfo.buff); // Get the password from the user
    if(strcmp(localinfo.buff,"thegeekstuff")) { // Check the password
        printf("\nWrong Password \n");
    } else {
        printf("\nCorrect Password \n");
        localinfo.pass = 1; // Set a flag denoting correct pass
    }

    if(localinfo.pass) { /* Now Give root or admin rights to user*/
        printf("\nRoot privileges given to the user \n");
    }

    return 0;
}

Lets test it!
Let's look inside...

• GDB is a common debugger for C programs.
• gcc –g –O0 –fno-stack-protector myFile.c
  • -g option enables debug symbols and thus debugging
  • -fno-stack-protector disables StackGuard protection
  • -O0 turns off optimization (makes debugging easier)

• DDD is a graphical front-end to GDB. Typically GDB is used from a command line interface

Coming up: Lessons
Lessons

• Buffer overflows occur when memory bounds of a variable are exceeded

• They can occur on the heap or the stack

• Buffer overflows have unintended consequences.

• There are protections against unsafe operations however they may not always be used (or known)

Coming up: References
References

• http://blog.zynamics.com/2010/03/12/a-gentle-introduction-to-return-oriented-programming/
• http://cseweb.ucsd.edu/~hovav/dist/geometry.pdf
• Stealth: http://www.suse.de/~krahmer/no-nx.pdf
• http://cseweb.ucsd.edu/~hovav/dist/rop.pdf