

Computer and Network Security

Homework and Programming Assignment 3

Dr. Edwin Sha

Due Date: 10am, April 24, 2012

Stack Overflow for SUN Sparc. (Total 130 points)

NOTE that there are non-programming questions in this assignment. Don't miss them. This can be done by a team. Please use C or C++ to write your programs. All the programs you wrote should be in ONE directory before you submit them electronically. I also need your report in hard copy. Each team please just submit one copy.

First read those articles posted in the course web page about stack overflow. Our homework will focus on SUN sparc machines because to exploit x86 stack-overflow is too easy. This is not just a programming assignment. **In your report, you must answer all the questions listed here in the report in addition to your programs.**

First you should do the following: copy all the files.

cp -rf /people/cs/e/edsha/security/2012S/hw3 .

See README, attack.c and server.c. See how the program "attack" can use stack overflow to run a shell after executing the program "server." Is it fun?

Imagine that the server program is a setuid program owned by the root, then after you penetrate the program with a shell running, you become the ROOT and you can remove the whole file system if you want. So it is extremely risky to have a setuid program with stack-overflow vulnerability.

- (10 points) This is for Sparc architecture. The example server program is very easy: main() calls copy().
 - Draw the layout of the stack frame before the main calls copy(), and the layout of stack frames after the main calls copy().
 - In the copy(), how it accesses its input parameter, "*a"?
 - Describe how a function call is made in Sparc machines in terms of stack frame and some special instructions. (suggestion: use google search to search "sparc stack frame" to collect information, and see how it is different from x86 machines.)You can draw a figure by hand to show the layout of stack frames of a Sun Sparc machine. Make sure that you indicate where the return address is stored.
- (5 points) In the example program, attack.c, why the new return address is set to be that value? Is it really the beginning of the shell code? If not, why this address can lead to the execution of the shell code?
- (5 points) Read article 2. How to increase the chance that the compromised new return address will jump to an address so the inserted code in the buffer can be successfully executed?
- (5 points) We know that exploiting stack overflow in a SUN sparc machine is harder than exploiting overflow in a x86 machine. Why? List 2 reasons.

5. Programming Part (90 points):

In the hw3 directory, there are three exploitable programs, server1, server2 and server3. I have given you the complete source code of server1.c and what server2.c and server3.c look like. Now you need to write programs, exploit1, exploit2 and exploit3 to exploit their stack overflow vulnerabilities respectively. All of your exploit programs should successfully launch a shell eventually.

Your program should be user friendly and general. I would suggest you to have at least two arguments into your program: buffersize and offset. For example, run your program something like

exploit1 buffersize offset

You can try your program A FEW times to find out the working arguments so you can set up a working buffer to overflow the stacks for server1 server2 and server3. After you find out what will be the working arguments from your general attack program, put the explanation in your README file and in your report. In your README file, you must let TA know how to run your programs to exploit server1, server2 and server3. **Make sure that your programs and parameters work for apache.utdallas.edu!! TA will run your programs on apache.**

6. (6 points) In server3, when copy2() is executed, there are three return addresses: copy2's return address that is used to go back to copy1, copy1's return address that is used to go back to main, and main's return address that is used to go back to its parent. Which return address is your exploit program replaced so the attack shell code can be executed.
7. (9 points) When you try to exploit a server, the results may be one of the following three cases: (1) the server program finishes okay. (2) abort with **illegal instruction**. (3) abort with **segmentation fault or bus error**. Please explain the causes of each case and what action you like to do when having each case in order to successfully exploit the target server program? You should explain them in terms of buffer size and return address that jumps to, etc.
8. SUGGESTIONS: You must read article 1 and article 2 to get some ideas first. My attack.c in hw3 is NOT a good one for you to do this assignment. A program in one article there can be a good one for you to revise to finish this assignment. Check it out.
- The programming must be done in SUN Sparc machines, for example, apache.utdallas.edu.
9. The grading of your programs is based on the clarity, correctness, efficiency, and programming styles of your programs. **TA will run your programs on apache.utdallas.edu**. So make sure your programs and parameters work correctly on apache.utdallas.edu. Do not expect that you can get full points even your program runs correctly.

Be honest. If your program does not work, SAY SO. TA will read your programs, compile your

programs and test your programs. If your program does not work but you "claim" yours works, you will have severe penalty!

How to submit your programs; Each team should submit a HARD COPY of your programs, AND an ELECTRONIC COPY of your programs before the due date. Just one copy for each team, please.

1. Go to the directory that contains all your source code, makefile, executables, README, required files, etc. You must explain your programs clearly in the README file. **You must write down your name and email address in the README file.**
2. Make sure the directory contains ALL the files that are able to be compiled and executed for the assignment. You must remove any redundant files so the tar file will not be too large! Let me say again. You must remove any unnecessary files before submission. The TA will use your Makefile to compile your program.
3. In that working directory, issue the command `~edsha/handin/handin.ass3` (Note the ~ in the beginning.) or `/people/cs/e/edsha/handin/handin.ass3` on "apache.utdallas.edu" or one of the school SUN machines. We will look at the time of the submission file to know if you submit your programs before or after the due date. This command must be issued in the directory that contains the files for the assignment.

No cheating is allowed. I will be very upset if we find any cheating. You must do the homework by yourself.