University Of Washington, CSE 590P – Computer Security - Homework 4

Tadayoshi Kohno, John Manferdelli

Due: 4:30pm February 8, 2007. This homework is worth 23 points.

See the course website (http://www.cs.washington.edu/education/courses/csep590b/07wi/) for instructions on how to submit your homework. For this assignment, you should submit a PDF file named 'YourLastName-YourFirstInitial-HW4.pdf. Please type your name on the first page of your assignment. You may also submit C source code for extra credit; the name of your source code should be sploit1.c. Your name should be in the leading comments of your source code.

1. (5 points total.) Explain why the NX (no execute) bit might help protect against buffer overflow attacks, but why the NX bit is, by itself, not a complete solution.

2. (2 points total.) The C function strcpy is unsafe. The C function strncpy is supposed to be better. Explain why strncpy is supposed to be better, and then explain why its usage may not be completely safe. For the latter part, you're free to suppose a situation (i.e., your answer might begin with "suppose that …").

3. (6 points total.) Besides strcpy mentioned above, list three other unsafe C functions and describe why they are unsafe. Then give the "safer" version of these functions (like strncpy is for strcpy). Then describe one problem that might still arise with each "safer" variant, or explain why no such problem exists.

4. (10 points total.) For this part of the assignment, you will have the opportunity to break into (obtain root access) on one of the University of Washington's Linux computers. Part of this assignment is based on Dan Boneh's and John Mitchell's CS 155 class at Stanford and Geoff Voelker et al's class at UCSD (co-taught at UW).

   4.1. Make sure that you can log into the machine umnak.cs.washington.edu. You will need to ssh into these machines (no telnet, etc). Your login should be the same as the one on your CSE accounts.

   4.2. From umnak.cs.washington.edu, make sure that you can ssh into fullsail.cs.washington.edu. (You will not be able to ssh into fullsail.cs.washington.edu from any other machine.) Fullsail.cs.washington.edu has a vulnerable suid root application that you are going to exploit; you can use umnak.cs.washington.edu as your development machine.) [UPDATE Feb 2, 2007: umnak.cs.washington.edu currently has address space randomization enabled; fullsail has it disabled. So, until support@cs disables address space randomization on umnak, please use fullsail for development.] [UPDATE Feb 4, 2007: You can now use umnak.]

   4.3. Download the HWclass4Source.tgz file from the course website, scp the files onto umnak, and extract the files (via "tar xzvf HWclass4Source.tgz"). There are several source code files in the extracted HWclass4Source directory:
      - Makefile --- this is the makefile.
      - exploit3.c --- this is the exploit3.c source code from Aleph One's article.
      - overflow1.c --- this is a slightly modified overflow1.c from Aleph One's article.
      - testsc2.c --- this is a slightly modified testsc2.c from Aleph One's article.
      - vulnerable.c --- this is a slightly modified vulnerable.c from Aleph One's article.
      - sploit1.c --- this is the template for exploit1 --- you will be changing this file.
      - target1.c --- this is application that you are going to exploit.

   4.4. Make sure that you can compile the programs in the HWclass4Source directory. You do this by typing "make" in that directory.

   4.5. (5 points.) Read Aleph One's article (link below). Experiment with the programs "testsc2" and "overflow1." Write and submit a short (up to two paragraphs) summary. [UPDATE Feb 4, 2007: Please write a short summary of what you learned from the article and your experimentation. If you learned the most from the article and not the experiments, then you can say that and summarize the article. On the other hand, if you learned a lot from the experiments, please also include a discussion of your experiments in the summary.]

4.6. (5 points. DUE 2/15/2007.) [UPDATE Feb 6, 2007: Please see enhanced description below.]

4.6.1. Try to use "exploit3" to mount a buffer overflow attack on "vulnerable". You can base your attack on the discussion in Aleph One's paper, but the parameters to "exploit3" will be different. For this portion of the assignment:

- (5 points.) If you are *not* successful, explain what you tried and what you tried and what the challenges were. You may find the Wiki discussion helpful.
- (5 points + 5 points extra credit.) If you *were successful*, explain the steps you went through to determine the input parameters to "exploit3." Include a transcript of a successful attack against the program "vulnerable" using "exploit3."
    - Here is my transcript, with xxx replacing some of the fields:

```
[yoshi@fullsail ~/HWclass4Source]$ ./exploit3 xxx xxx
Using address: 0xxxx
bash-3.1$ ./vulnerable $EGG
main at address 0xxxx
sh-3.1$ exit
```

  You will also get full credit if you modify exploit3.c. If you do this, please include your exploit3.c in your submission and explain what your modifications do and why they work.
- (5 points extra credit.) This extra credit problem may help some of you with the above extra credit problem. Download the new exploit3-new.c file from the course website. Read the comments at the top of exploit3-new.c and figure out the value for MY_DIFF. Include in your submission your value a transcript of a successful run, and explain how/why your attack worked. Hint: Where does the *address* of buf appear in the stack? Also, the header of exploit3-new.c contains some notes on helpful gdb commands.

You are to use the machine umnak.cs.washington.edu for this task. [Use fullsail instead of umnak for now; see update above.] [UPDATE Feb 4, 2007: You can now use umnak.]

4.7. (10 points extra credit. DUE 2/15/2007.) You are now to modify "sploit1.c" so that the compiled "sploit1" program mounts a buffer overflow on target1.

- The first step is to develop your exploit "sploit1" on umnak.cs.washington.edu. [Use fullsail instead of umnak for now; see update above.] [UPDATE Feb 4, 2007: You can now use umnak.]
- The second step is to copy your executable for sploit1 into the /tmp directory on fullsail.cs.washington.edu. Your goal here is to obtain root by attacking the /tmp/target1 suid root binary on fullsail.cs.washington.edu.
    - You should expect to see something like this:

```
[yoshi@fullsail /tmp]$ whoami
yoshi
[yoshi@fullsail /tmp]$ ./sploit1
sh-3.1# whoami
root
sh-3.1# exit
```

- For this portion of the assignment, you are to:
    - Submit your source code via the Catalyst Tools (name your file sploit1.c, we will run it against a target1 file in the same directory).
    - Email us the contents of the /etc/secret file on fullsail.cs.washington.edu (you can only read this file as root).
- Once you obtain root on fullsail.cs.washington.edu, you are not to do anything besides execute the command "whoami" (to verify that you are root) and "cat /etc/secret" (to output the contents of the "secret" file).

Reading:

Aleph One, "Smashing The Stack For Fun And Profit," http://insecure.org/stf/smashstack.html

Problem 4 Related Assignments (you can find additional related works, slides, and other helpful information for Problem 4 here):

Stanford CS 155 Project 1: http://crypto.stanford.edu/cs155/

UCSD CSE 291: http://www.cse.ucsd.edu/users/voelker/cse291/fa05/redteam.html