CS5434 Homework 1: Vulnerabilities

In class, you've learned about some of the terrifying consequences of writing insecure code. For this homework, you will gain some practical experience in exploiting computer vulnerabilities.

You will need:

1. A partner-in-crime (find one, steal one, but do not share one! Criminals have principles too, and academic integrity policies apply). Use Piazza if you are lonely and need help with matchmaking.

2. VMWare Workstation v10. Download through the Cornell University CIS academic alliance website below. You should have received login credentials for this website from a CIS e-mail sent at the start of the semester.

   https://e5.onthehub.com/d.ashx?s=p6mvdgohev

3. A copy of the HW1 virtual machine, which contains everything you'll need for homework 1. This can be found at

   - http://www.cs.cornell.edu/courses/cs5434/2015fa/HW1.zip

4. Lots of patience and determination (so please start on your homework early).

5. To read the following important notice.

Important notice:

Although this homework takes you through the basics of exploiting computer vulnerabilities, the exploitation of (or the act of attempting to exploit) any equipment for which you do not exercise complete ownership can land you in very serious trouble. Be careful to keep all your activity confined to the virtual machine we have set up for you.

Instructions on the virtual machine:

1. Unzip the virtual machine someplace on your computer.

2. Upon starting VMware Workstation, click on File… Open. Proceed to open the virtual machine named "HW machine" as found in the unzipped folder.

3. VMware will ask if you moved or copied the machine. Click on "copied".

4. Power on the machine.

5. The login password is "**acorns**" (without the quotes). This is the same password if you require sudo access.

Here's how you would access the homework:

Launch terminal and cd into the HW1 directory. There, you will find the source code to a sophisticated banking program, innocuously named bank.c. Open it in your favorite code editor (we've generously preinstalled vim and emacs but feel free to install something else if you want to) and study it with your partner.

Next, build the program and try it out. To simplify your work, we've supplied a Makefile that contains all the necessary flags to disable stack guards and to enable code execution on the stack. It also suppresses a number of warnings so the compilation output is not obnoxiously verbose. To build the code, simply type "make". Executing the code is as simple as typing "./bank". root privilege is not required.

Here's a quick explanation on how the banking program works.

Upon startup, the user is presented with a login screen. If you've studied the source code, you'll notice that there are 5 predefined user accounts. Logging in is as simple as entering a username and supplying the corresponding password. Although account information is 'hard coded' into the source, assume that users won't have access to the code. You can make the following assumptions:

1. Users know their own usernames and passwords.

2. Users know other usernames (so they can execute bank transfers), but not the passwords to those accounts.

3. The 'exit' username is special and accepts any password. It is primarily used to terminate the program at the end of the day.

After logging in, the banking system allows a user to change his password or transfer money between accounts. Once a day, after the bank closes, a trustworthy employee is sent to shut down the program via the 'exit' account. The employee types the special 'exit' userid at the login prompt, which causes the program to terminate after dumping financial and password records into files. Financial records are stored in 'userid.txt' files, so for example, zteo's closing balance will be stored in 'zteo.txt'. Correspondingly, password records are stored as 'userid_pwd.txt' files, so 'zteo_pwd.txt' contains the plaintext password for the 'zteo' account at the time of closing. After the files are dumped, the same trustworthy employee inspects the files and manually updates the source code of the bank program to reflect balance and password changes for the next business day (yucks… but that's the state-of-the-art in banking).

Here is your assignment:

1. Review the supplementary slides on how to use gdb.

2. Together with your partner, attempt to discover and exploit as many security vulnerabilities as you can spot, with a focus on the exploits discussed in class.

3. For each vulnerability that you exploit, create a .bin file that, when redirected to stdin of the bank program, demonstrates the exploit. Name the exploit sequentially (so exploits should be named 1.bin, 2.bin, 3.bin, etc.). We will grade these exploits under gdb.

4. In the supplied README file, fill out your names, netIDs and answers to the questions in the file.

5. When you are done with the homework, prepare your submission by zipping up only the .bin files and the README file, then upload your zip file via CMS. (Protip: launch Firefox directly from within the virtual machine so you don't need to scp/copy the files out of your VM to your host machine).

Please help keep the homework fun! Don't discuss your techniques or exploits with other students in the class (unless of course he/she is your partner). As a solemn reminder, academic integrity policies apply and we are very strict about enforcing them.

If you have any questions, please feel free to post them onto Piazza. You are also welcome to utilize our office hours should you have questions you'd like to ask in person.