Project 1

Due: September 22

Goal

1. The goal of this assignment is to gain hands-on experience with the effect of buffer overflow, format string, and double free bugs. All work in this project must be done on the virtual machine provided on the course website. You will need to download VirtualBox from https://www.virtualbox.org/wiki/Downloads.

2. You are given the source code for seven exploitable programs (/tmp/target1, /tmp/target2). These programs are to be installed as setuid root in the VMware virtual machine. Your goal is to write two exploit programs (sploit1, sploit2). Program sploit[i] will execute program /tmp/target[i] giving it certain input that should result in a root shell on the VMware virtual machine..

3. The skeletons for sploit1, sploit2 are provided in the sploits/ directory. Note that the exploit programs are very short, so there is no need to write a lot of code here.

The Environment

1. You will test your exploit programs within a VMware virtual machine. To do this, you will need to download the virtual machine image provided on the course website as well as VirtualBox . VirtualBox can run on Linux, Mac OS X, and Windows, and is freely available.

2. The virtual machine we provide is configured with Debian Etch. We've left the package management system installed in the image, so should you need any other packages to do your work (e.g. emacs), you can install it with the command apt-get (e.g. apt-get install emacs).

3. The virtual machine is configured to use NAT (Network Address Translation) for networking. From the virtual machine, you can type ifconfig as root to see the IP address of the virtual machine. It should be listed under the field inet addr: under eth0.

4. The virtual machine also has an ssh server. You can ssh into the vm (virtual machine) from your machine, using the IP address produced by ifconfig (as above) as the destination. You can use this to transfer files onto the virtual machine using scp or an sftp client like SecureFX. Alternatively, you can fetch files directly from your own website on the virtual machine using wget.

The Targets

1. The targets/ directory in the assignment tarball contains the source code for the targets, along with a Makefile specifying how they are to be built.

2. Your exploits should assume that the compiled target programs are installed

setuid-root in
/tmp – /tmp/target1, /tmp/target2, etc.

The Exploits

The sploits/ directory in the assignment tarball contains skeleton source for the exploits which you are to write, along with a Makefile for building them. Also included is shellcode.h, which gives Aleph One's shellcode.

The Assignment

You are to write exploits, one per target. Each exploit, when run in the virtual machine with its target installed setuid-root in /tmp, should yield a root shell (/bin/sh).

Hints

1. Read Aleph One's "Smashing the Stack for Fun and Profit." Carefully. Draw diagrams so that you have a good understanding of what happens to the stack, program counter, and relevant registers before and after a function call. Other important papers/resources you should read/skim include:

   (a) scut's - "Exploiting Format String Vulnerabilities"

   (b Intel - Intel Architecture Guide for Software Developers

   All the papers should be linked from SOCS or easily found through Google. It will be helpful to have a solid understanding of the basic buffer overflow exploits before reading the more advanced exploits.

2. gdb is your best friend in this assignment, particularly to understand what's going on. Specif- ically, note the "disassemble" and "stepi" commands. You may find the 'x' command useful to examine memory (and the different ways you can print the contents such as /a /i after x). The 'info register' command is helpful in printing out the contents of registers such as ebp and esp.

   A useful way to run gdb is to use the -e and -s command line flags; for example, the command 'gdb -e sploit2 -s /tmp/target2' in the vm tells gdb to execute sploit2 and use the symbol file in target2. These flags let you trace the execution of the target2 after the sploit has forked off the execve process. When running gdb using these command line flags, be sure to first issue 'catch exec' then 'run' the program before you set any breakpoints; the command 'run' naturally breaks the execution at the first execve call before the target is actually exec-ed, so you can set your breakpoints when gdb catches the execve. Note that if you try to set break points before entering the command 'run', you'll get a segmentation fault.

   If you wish, you can instrument your code with arbitrary assembly using the _asm_() pseud-ofunction.

3. Make sure that your exploits work within the provided virtual machine.

4. **Start early**. Theoretical knowledge of exploits does not readily translate into the ability to write working exploits. Target1 is relatively simple and the other problems are quite a bit more complicated.

Warnings

Aleph One gives code that calculates addresses on the target's stack based on addresses on the exploit's stack. Addresses on the exploit's stack can change based on how the exploit is executed (working directory, arguments, environment, etc.); in my testing, I do not guarantee to execute your exploits as bash does.

You must therefore hard-code target stack locations in your exploits. You should *not* use a function such as get sp() in the exploits you hand in.

Deliverables

1. You are to provide a tarball (i.e., a .tar.gz or .tar.bz2 file) containing the source files and Makefile for building your exploits. All the exploits should build if the "make" command is issued.

2. There should be no directory structure: all files in the tarball should be in its root directory.
   (Run tar from inside the sploits/ directory.)

3. Along with your exploits, you must include file called ID which contains, on a single line, the following: your your username; and your name, in the format last name, comma, first name. An example:

   ```
   $ cat ./ID
   hermann Buhl, Hermann
   $
   ```

   If you did the project with a partner, then both of you will submit only one solution and the ID file will have two lines giving the relevant information.

4. Please submit the tarball to CANVAS. Again, make sure that you test your exploits within the provided virtual machine.

How to set up the Environment

Here are the following steps that I take to set up the environment.

1. Download and install VirtualBox from https://www.virtualbox.org/wiki/Downloads. (for Windows, Linux, Mac OS X).

2. Download the virtual machine tarball (box.tar) from CANVAS.

3. Decompress the virtual machine tarball, then open the file box.vmx using

VirtualBox. If VirtualBox asks you if you moved or copied the virtual machine, say that you copied it.

4. Login to the virtual machine. There are two accounts, root with the password root, and user with the password user.

5. Ensure that networking is working by typing ifconfig and checking that the inet addr: field of eth0 has a valid IP address. Make sure you can reach the machine by attempting to ssh into it.

6. Download the project 1 tarball (pp1.tar) onto the virtual machine. You can do this by downloading the tarball first, then using scp or an sftp client to transfer the files onto the vm.

7. Copy the sploits/ directory to the user's home directory (and make sure to set the owner- ship so that user can access them 'chown -R user:user sploits'), and the contents of the targets/ directory to /tmp. In /tmp, make the targets then apply the following commands to set up the permissions so that the targets are owned by root, are setuid root, and the .c files are publicly readable. Doing this correctly will allow gdb to read the source files while you are debugging.

box:/tmp# chown root:root target? ; chmod 4755 target? ; chmod a+r target?.c

8. Everytime you reboot the vm, you'll have to set up the targets in vm's /tmp because it'll be wiped clean.