CSE 127: Computer Security **Spring 2010**

Project #1

Part 1 Due: Sunday, April 11th, 2010, 11:59 PM
Part 2 Due: Sunday, April 18th, 2010, 11:59 PM

Goal

The goal of this assignment is to gain hands-on experience with the effect of buffer overflow and other memory-safety bugs.

All work in this project must be done on the VMware virtual machine provided on the course website; see below for information about this environment.

You are given, in the `targets/` directory, the source code for five exploitable programs, `target1.c`, ... `target5.c`. These programs are to compiled and installed, setuid root, in the `/tmp` directory of your VM. Your goal is to write five exploit programs `sploit1`, ..., `sploit5`, each of which will execute the corresponding target with input that exploits that target's bug, giving a root shell on the VM.

We have provided skeletons for these exploits programs in the `sploits/` directory, as `sploit1.c`, ..., `sploit5.c`. Our own solutions, incidentally, are very short: just 200 or 300 lines in total. So while understanding and exploiting the bugs will not be easy, you will not need to write a lot of code.

Collaboration

You may work on this project with one other person. You will turn in a single set of solutions together, but it is expected that both of you understand and can explain how to exploit each target. You must not discuss the project with anyone in the class besides your partner, nor with anyone outside of class. (For more about the collaboration policy and academic integrity, see the class syllabus.)

The Environment

You (and we, for grading!) will test your exploit programs within a VMware virtual machine. To use this VM on your personal Windows or Linux machine, you will need to download the virtual machine image provided on the course website — `boxes-2.1.tar.bz2` — as well as the free VMware Player from VMware's website.

We have also arranged for VMware Player and VMware Workstation to be installed on the `ieng6` machines in the B230 cluster in the basement of EBU 3B.

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`.

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

The networking setup in B230 slightly restricts outgoing network connections from the VM, but this should not pose any problems.

The Targets

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.

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. Also read the "suggested reading" listed in the project README. You will want to have a good understanding of what happens to the stack, program counter, and relevant registers before and after a function call. Read scut's "Exploiting Format String Vulnerabilities," linked from the course syllabus. It will be helpful to have a solid understanding of the basic buffer overflow exploits before reading the more advanced exploit papers.

2. The `gdb` debugger is your best friend in this assignment, as you'll want to understand what's going on in the target program's memory space. Specifically, 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` or `/i` after `x`). The "`info register`" command is helpful in printing out the contents of registers such as `ebp` and `esp`.

A useful command to run gdb is to use the `-e` and `-s` command line flags; for example, the command "`gdb -e ./sploit3 -s /tmp/target3`" tells gdb to execute `sploit3` and use the symbol file in `target3`, which allows you to trace execution in the target in exactly the way it is executed by the exploit. By contrast, executing just "`gdb /tmp/target3`" will trace the target in the way it is executed by the shell, so the memory layout will be different. (Avoiding a segfault while using this requires careful timing in setting breakpoints. See the assignment README for the details.)

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. The first target is relatively simple to exploit, but the difficulty ramps up from there . . .

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 our testing, we do not guarantee to execute your exploits the same way 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

To encourage you to start on the project early, part 1 (due on April 11th, 11:59 PM) consists of `target1` and `target2`. Part 2, due one week later, consists of the other three targets.

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.

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

Along with your exploits, you must include file called ID which contains, on a single line, the following: your UCSD 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.

You may want to include a README file with comments about your experiences or suggestions for improving the assignment.

Instructions for submitting the tarball will be posted on the course website. Again, make sure that you test your exploits within the provided virtual machine.

How to set up the Environment

Your TA, Albert Park, will be showing you how to set up the Boxes environment in the section on Friday, April 2nd. Below are some tips if you'd like to get started early.

Setting up Boxes on your own machine. Here are the steps needed to set up the environment on your own machine.

1. Download and install VMware player from `http://www.vmware.com/products/player/` (for Windows and Linux) or VMware Fusion from `http://www.vmware.com/products/fusion/` (for Mac OS X).
2. Download the VMware virtual machine tarball (boxes-2.1.tar.bz2) from the course website.
3. Decompress the virtual machine tarball, then browse to the directory `Boxes2.vmwarevm` using the `Open an existing Virtual Machine` option in VMware Player and select the file `Boxes2.vmx`. If VMware Player asks you if you moved or copied the virtual machine, say that you *moved* it.
4. Log in 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 from the physical host on which you are running VMware.
6. Download the project 1 tarball (pp1.tar.gz) 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. Alternatively, log in as user to the vm and type

   ```
   wget https://cseweb.ucsd.edu/classes/sp10/cse127/pp1.tar.gz
   ```

7. Unpack the pp1 tarball in the user's home directory. Use the Makefile to build the targets in `~/pp1/targets/` and install them in `/tmp` ("`make; make install`"). When you are ready to test that your exploits work with setuid targets you can run "`make setuid`" in the `~user/pp1/targets/` directory, *as root.*
8. Everytime you reboot the VM, you'll have to set up the targets in the VM's `/tmp` directory because it'll have been wiped clean.

Using Boxes on the B230 lab machines. If you want to use the machines in the B230 lab, located in the basement in building EBU 3B, here is what you need to do.

1. Log in to any machine under Red Hat Linux using your CSE 127 username and password (if you don't know this information go to `https://sdacs.ucsd.edu/~icc/index.php`).

2. Extract the boxes virtual machine file,

   ```
   /home/linux/ieng6/cs127s/public/boxes2.1-ieng6.tar.bz2
   ```

   to your home directory. (The home directory of a user with CSE127 username ***student*** is `/home/linux/ieng6/cs127s/`*`student`*.) Do not copy the tarball to your home directory first and then extract it because you will exceed the disk quota assigned to your account!) Then download the project 1 tarball (pp1.tar.gz) from the course website onto your ieng6 account.

3. Open the VMware Workstation by going to Applications→ System Tools→ VMware workstation; then browse to the `Boxes2.vmwarevm` directory (located in your home directory) using the `Open an existing Virtual Machine` option and select the file `Boxes2.vmx`. If VMware Workstation asks you if you moved or copied the virtual machine, say that you *moved* it.

4. Log in to the virtual machine. Again, there are two accounts, `root` with the password `root`, and `user` with the password `user`. You can ssh into your VM from the host (i.e., the B230 lab machine on which you are running VMware) by using the command "`ssh -l user 192.168.1.128`", and can use scp or sftp the same way.

5. Transfer the project 1 tarball (pp1.tar.gz) from your ieng6 account onto the VM using the `scp` command.

6. Unpack the pp1 tarball in the user's home directory. Use the Makefile to build the targets in `~/pp1/targets/` and install them in `/tmp` ("`make; make install`"). When you are ready to test that your exploits work with setuid targets you can run "`make setuid`" in the `~user/pp1/targets/` directory, *as root*.

7. Everytime you reboot the VM, you'll have to set up the targets in the VM's `/tmp` directory because it'll have been wiped clean.