CS 345 - Programming Languages
Fall 2010

Homework #2

Due: 2pm CDT (in class), September 30, 2010

Collaboration policy

This assignment can be done in **teams at most two students**.

Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Science Department Code of Conduct can be found at `http://www.cs.utexas.edu/academics/conduct/`.

Late submission policy

This homework is due at the **beginning of class** on **September 30**. All late submissions will be subject to the following policy.

You start the semester with a credit of 3 late days. For the purpose of counting late days, a "day" is 24 hours starting at 2pm on the assignment's due date. Partial days are rounded up to the next full day. You are free to divide your late days among the take-home assignments any way you want: submit four assignments 1 day late, submit one assignment 3 days late, *etc.* After your 3 days are used up, no late submissions will be accepted and you will automatically receive 0 points for each late assignment.

Homework #2 (35 points + 15 bonus points)

Objective

The objective of this project is to give you hands-on experience with implementing buffer overflow exploits. You are given the source code for three exploitable programs (`target1.c`, `target2.c`, `target3.c`). These programs are all installed with root (*i.e.*, superuser) privileges in the the VMware virtual machine.

Your goal is to write two exploit programs (you only need to write the third exploit if you want 15 bonus points). Each exploit program will execute its target, giving it a certain input that should result in a root shell on the VMware virtual machine. See below (Your Assignment) for more details.

Files

You will need:

- The VMware Player:
  `http://www.vmware.com/products/player/`
- The virtual machine image:
  `http://www.cs.utexas.edu/~shmat/courses/cs345/box.tar.bz2`
- The project files:
  `http://www.cs.utexas.edu/~shmat/courses/cs345/cs345-hw2.tar.bz2`

VMware environment

You will test your exploit programs in a VMware virtual machine. To do this, you will need to download the virtual machine image as well as the VMware Player from VMware's website (see above). VMware Player can run on Linux, Windows, and Mac OS X (VMware Fusion).

The virtual machine we provide is configured with Debian Etch. 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`). [1]

The virtual machine is configured to use NAT 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 virtual machine from your machine, using the IP address produced by `ifconfig` (see above) as the destination. You can also use this to transfer files onto the virtual machine using `scp`. Alternatively, you can fetch files directly from the web on the VM using `wget`.

[1] You may need to edit the `/etc/apt/sources.list` file and replace `http://mirrors.kernel.org/debian` with `http://archive.debian.org/debian` everywhere.

Targets

The project files (`cs345-hw2.tar.bz2`) contain 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.*

Exploits

The project files (`cs345-hw2.tar.bz2`) also contain skeleton source code 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. Exploit programs are very short, so there is no need to write a lot of code.

Your assignment

You are to write one exploit 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`). You can use `whoami` to tell if you are root or not.

Grading

There are two targets. Each successful exploit will give you a certain amount of points:

Target 1: 20 points

Target 2: 15 points

Target 3 is the **bonus** target. It is worth 15 extra points on top of the regular points for this assignment.

Hints

Read Aleph One's "Smashing the Stack for Fun and Profit" **carefully**. Read scut's "Exploiting Format String Vulnerabilities." Both are linked from the reference section of the course website.

`gdb` is your best friend in this assignment. It will help you inspect the contents of memory as your target is executing and generally understand what's going on. In particular, notice 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`. Another very useful command is `info frame`. It prints a detailed description of the selected frame.

When you run `gdb`, you will find the `-e` and `-s` command-line flags useful. For example, the command `gdb -e sploit1 -s /tmp/target1` in the virtual machine tells `gdb` to execute `sploit1` and use the symbol file in `target1`. These flags let you trace the execution of `target1` 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 `gdb` has trouble finding the source files of targets, try running it with the `-d /tmp` command-line flag.

If you wish, you can instrument your code with arbitrary assembly using the `__asm__()` pseudofunction.

IMPORTANT: Your code **must** run within the provided virtual machine environment.

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 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.

Your exploit programs should not take any command-line arguments.

Deliverables

You will submit a single tarball that contains the source code for all your exploits, along with any files (`Makefile`, `shellcode.h`) necessary for building them.

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).

You will submit your files using the `turnin` command on the UTCS system. The grader is `austin` and the homework name is `buffer`.

The tarball should also include a file `SUBMISSION`. The first line should state how many late days were used (if any). Then give the following on a single line, one for each student:

- your UTEID
- your UTCS username
- your real name

You may want to include a `README` file with comments about your experiences or suggestions for improvement.

Getting Started

Please make sure you start early. This ensures you can set up the proper VM environment in which to write your exploits.

1. Download and install VMware Player (Windows and Linux) or VMware Fusion (Mac OS X: `http://www.vmware.com/download/fusion/`).
2. Download the VMware virtual machine tarball (`box.tar.bz2`).
3. Decompress the virtual machine tarball, then open the file `box.vmx` using VMware Player. If VMware Player 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/root and user/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 homework files (`cs345-hw2.tar.bz2`) onto the virtual machine. You can do this by downloading the tarball first, and then using `scp` to transfer the files onto the VM. Alternatively, you can log in as root to the VM and use `wget`.
7. Copy the sploits directory to the user's home directory (and make sure to set the ownership so that user can access them "`chown -R user:user sploits`"), and the target directory to the root's home directory. Make the targets and copy the targets to `/tmp` together with the corresponding .c files. Set up the permissions so that the targets are owned by root, are setuid root, and the .c files are publicly readable:

   ```
   chown root:root target?  ; chmod 4755 target?; chmod a+r target?.c
   ```
8. Every time you restart the VM, you'll have to set up the targets in the VM's `/tmp` because it'll be wiped clean.