CS155: Computer Security **Spring 2004**

Project #1

Due: Thursday, April 22, 2004.

Goal

The goal of this assignment is to gain hands-on experience with the effect of buffer overflow bugs and format string bugs. All work in this project must be done on a system called boxes (implemented using User-Mode Linux) available on the course web site.

You are given the source code for seven exploitable programs (`/tmp/target1`, ... , `/tmp/target7`). These programs are all installed as setuid root in the boxes system. Your goal is to write seven exploit programs (`exploit1`, ..., `exploit7`). Program `exploit[i]` will execute program `/tmp/target[i]` giving it certain input that should result in a root shell on the boxes system.

The skeletons for `exploit1`, ..., `exploit7` are provided in the `exploit/` directory. Note that the exploit programs are very short, so there is no need to write a lot of code here.

The Environment

You will test your exploit programs within a system called Boxes. Boxes, based on User-Mode Linux, allows you to boot a fully-functional Linux system as a userland process on another Linux machine.

Boxes is available from the course website. It should run on x86 GNU/Linux machines running a recent 2.4-series kernel. You must install Boxes on a GNU/Linux machine on your own.

Please refer to the README file in the Boxes distribution.

It is recommended that you test your exploits in a virtual machine booted with a "closedbox" kernel, so that you cannot accidentally damage your host account.

You can use the ssh daemons running in the image to transfer files from openboxes (with hostfs access) to closedboxes.

It is recommended that you develop your code on the host machine, or at least keep frequent backups. The User-Mode Linux kernel is mostly stable, but can occasionally crash.

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 Boxes environment with its target installed setuid-root in /tmp, should yield a root shell (/bin/sh).

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 course website.)

To understand what's going on, it is helpful to run code through gdb. In particular, notice the "disassemble" and "stepi" commands. You can instrument your code with arbitrary assembly using the __asm__() pseudofunction.

make sure that your exploits work within the Boxes 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 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

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 SUID number; your Leland username; and your name, in the format last name, comma, first name. An example:

```
$ cat ./ID
3133757 binky Clown, Binky The
$
```

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 Boxes environment.