Projects 1 and 2 : Control Flow Hijacking

CS 645: Network Security
Project 1 Due: April 18, 2012 11:59 pm EST
Project 2 Due: May 2, 2012 11:59 pm EST

April 4, 2012

This project is based on a project designed by Dan Boneh for his class. Thanks Dan.

Goal

1. The goal of these assignments is to gain hands-on experience with the effct of buffer overflow, format string, and double free bugs. All work in this project must be done on the VMware virtual machine provided on the course website. You can either download VMware Player from http://www.vmware.com/products/player/ or access the vm on tux using virt-run cs475-sploit.

2. You are given the source code for seven exploitable programs (`/tmp/target1, ... , /tmp/target7`). These programs are to be installed as setuid root in the VMware virtual machine. Your goal is to write seven exploit programs (`sploit1, ..., sploit7`). 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, ..., sploit7` 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.

4. **Project 1** is to write sploits for targets 1 and 2, (sploit1.c and sploit2.c). Project 1 is to be done individually.

5. **Project 2** is to write sploits for at least four of the remaining spoloits (sploits 3-7). These sploits, especially 5-7, are much harder than the first group. This project is to be done in groups.

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 VMware Player from VMware' website. VMware player can run on Linux, Mac OS X (VMware Fusion), and Windows, and is freely available.

   You can also access the virtual machine on tux using the virt-run command:

   ```
   virt-run cs475-sploit
   ```

   It is important to realize that when you exit the virtual machine it will return to the state you found it. Therefore, you must save all your work in an external location. I recommend using a version control system like git or svn. If you are accessing tux remotely, you should ssh using the -X option to forward your x-session.

2. The virtual machine we provide is congured 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, vi, screen), 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 from the 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 like SecureFX. Alternatively, you can fetch filees directly from the web on the vm 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.

Hints

1. Read Aleph One's "Smashing the Stack for Fun and Profit." Carefully. Read scut's "Exploiting Format String Vulnerabilities." All the papers are linked from the course syllabus and there are additional helpful readings in the README in the project tarfile. 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. 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 /i after x). The 'info register' command is helpful in printing out the contents of registers such as ebp as 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' in the vm tells gdb to execute sploit3 and use the symbol file in target3. These flags let you trace the execution of the target3 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 () pseudofunction.

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

3. 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. **Both Projects.** You will need to submit the source code for your exploits, along with any files (Makefile, shellcode.h) necessary for building them. The source code should be emailed to me in a tarball (tar.gz or tar.bz2) containing the source and Makefile for building your exploits. All the exploits should build if the "make" command is issued.

2. **Project 1.** You are responsible for providing exploits for the first two exploits. This project must be done and turned in individually.

3. **Project 2.** You are responsible for providing exploits for four of the remaining exploits. You will get extra credit for exploiting all seven and/or for providing an exploit for target-ec.

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

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

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

   For Project 2, the ID file should contain a line for each of you. You may work in groups of up to three.

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

7. If you attempted the extra credit and are submitting an exploit, please note this in your README.

8. Again, make sure that you test your exploits within the virtual machine. If they don't work, you won't get credit. Life is too short.

9. Late policy: Your grade will be reduced by 20% for each day the project is late.

How to set up the Environment

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/download/fusion/ (for Mac OS X).

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 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 (cs475-pp1.tar.bz2) 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 root to the vm and type//

   ```
   box:~# wget http://www.cs.drexel.edu/~greenie/cs475/projects/cs475-pp1.tar.bz2
   ```

7. Copy the sploits dir 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 target dir to root's home directory. Make the targets and copy the targets to /tmp together with the corresponding .c files. Using the following commands, set up the permissions so that the targets are owned by root, are setuid root, and the .c files are publicly readable.

```
box:~# 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.