# SIL765: Assignment 1 on Buffer Overflow Attacks

## Goal

1. The goal of this assignment is to gain hands-on experience with the effect of buffer overflow, integer overflow, format string, and double free bugs. All work in this project must be done on the VMware virtual machine provided below. You will need to download VMware Player.
2. Download VMware Player 3.1.4-385536 x86_64 bundle for 64 bit systems and VMware-Player-3.1.6-744570.i386 bundle for 32 bit systems. (for latest version download from VMware website)
3. 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.
4. 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.

## 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's website. VMware player can run on Linux, Mac OS X (VMware Fusion), 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., vim, emacs), you can install it with the command apt-get (e.g., apt-get install vim)
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". Alternatively, you can fetch files directly from the wen

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. The exploits should assume that the compiled target programs are installed setuid-root in /tmp. i.e., /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 two optional handouts - 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.". All the papers are 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 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 and esp. A useful command to run gdb is to use the -e and -s command line flags; for example, the command 'db -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 ()-pseudo function.
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.) 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.

## Extra Credit

There is an extra credit problem included in this project, target-ec.c. Submit this in order to get extra credit.

## Relevant Materials

1. Aleph One - Smashing the Stack for Fun and Profit
2. blexim - Basic Integer Overflows
3. scut/team teso - Exploiting Format String Vulnerabilities
4. anonymous - Once upon a free()
5. c0ntex - How to hijack the Global Offset Table with pointers for root shells
6. Intel - Intel Architecture Guide for Software Developers

## Files

- pp1-tarball: pp1.tar.bz2
- Virtual machine image: box.tar.bz2
  - Untar it.
  - Type "vmplayer box.vmx"
  - login="root", password="root"

## Submission Instructions

You only need to submit a tarball of the sploits/ directory. You will need to copy

your sploits/ directory out of the VM.

# Note:

1. To be done individually.
2. The last date of submission will be announced later.

## 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 (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.
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.

---