Jake Crane

4/16/2015

For this project I chose to use Ubuntu 14.10 (32-bit) because it is my preferred distribution of linux.

The first thing I had to do to for this project was disable Address space layout randomization (ASLR). Disabling ASLR makes stack smashing far more reliable.

I disabled ASLR by using the following command: sudo sysctl -w kernel.randomize\_va\_space=0

I also installed zsh to compare running the exploit using bash and zsh.

I accomplished this by using the following command: sudo apt-get install zsh

The next thing I did was remove the symbolic link located at /bin/sh and replace it with a link that pointed to /bin/zsh. I did this because the shellcode that was provided to me executes /bin/sh and I wanted to test the zsh shell. Later on I also experimented with changing the link back to /bin/bash.

I have included a shell script(test\_exploit.sh) to disable ASLR, install zsh, point /bin/sh to /bin/zsh, compile the source code, set stack's file permissions, run exploit and run stack. In my tests this script allowed me successfully smash the stack and gain full root access using my project on a fresh install of Ubuntu 14.10 (32-bit) and Linux Mint 17.1 (32-bit) without any prior configuration.

**Task 1:**

For the first task I modified the exploit program to store NOPS, the provided shell code and memory addresses, that were found to work through trial and error, into memory. Next, I compiled the exploit program using the following command: gcc -o exploit exploit.c -std=c99

I then ran exploit to create badfile.

Next, I compiled the stack program using the following command:

sudo gcc -o stack stack.c -zexecstack -fno-stack-protector

Then I set the stack file's permissions to -rwsr-xr-x so users have the ability to run the program with an euid of 0. I did this using the following command: sudo chmod 4755 stack

I then ran the stack program and was presented with the zsh prompt. By using the id command I was able to verify that I had obtained an euid of 0(root).

The next thing I did was run my setuid program that calls setuid(0) and spawns a new shell. By running the id command in this new shell I was able to verify that my uid was now 0(root).

**Task 2:**

In my tests, bash did not seem to behave any differently than zsh. I was able to get full root access (uid=0) using both shells. I verified this by using the id command.

**Task 3:**

When I enable address randomization and attempt to smash the stack I almost always get a Segmentation fault. If I run the following command, a shell is eventually spawned, but it can take quite some time.

command: sh -c "while [ 1 ]; do ./stack; done;"

**Task 4:**

When I compile and run stack without disabling the stack protection, I get the following message:

\*\*\* stack smashing detected \*\*\*: ./stack terminated

./test\_exploit.sh: line 21: 7470 Aborted (core dumped) ./stack