Name: Jianan Luo
UserId: j43luo
Student #: 20523403

Sploit1:

Vulnerability: Buffer overflow vulnerability

How to exploit:

At backup.c, line 231, the program is calling strcpy, main function basically allocate a buffer size 200 on to stack right after main return address && base pointer.
The stack should be look like following:

STACK1:

0xff.. -> functionName -> parameters -> returnAddress -> basePointer -> buffer(size200) -> .. 0x00..

It's not checking for input length, so when its calling strcpy, it will copy the fileName block into memory address where the buffer start, if it's over than 200, it will keep overwrite towards to 0xfffff.
It might cause the main function return address gets overwritten.
I created a payload with size 208, which has some \x90 (NOP) in front, follow by the shell code, and rest are all the addresses point to those \x90. The stack should be look like following:

STACK2:

0xffff.. -> functionName -> parameters -> addressPointsToNOP .... addressPointsToNOP -> shellCode -> NOP...NOP -> .. 0x00

How this will behave is when the program runs, the stack will get overwrite from STACK1 to STACK2 after calling strcpy, when it reach to the addressPointsToNOP it will jump to the address where NOPs are stored since the main return address had been overwritten already. NOP will make the pointer goes up until to hit shellCode. Then the shellCode will get executed.

How to fix:

Check size/length of the buffer before pass it in will perfectly solve exploit like this.

Sploit2:

Vulnerability: Bad code vulnerability

How to exploit:

At backup.c, line 163, the program is calling run_cmd("ls", "-la", BACKUP_DIRECTORY, NULL);
If we create a file called ls, it will still run the command, instead, it will treat ls as a executable. And run whatever we put in that file. Ex. /bin/sh

How to fix:

Provide an absolute path for ls. Use /bin/ls instead of ls on that line