cat narnia3.c

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \*\*argv){

int ifd, ofd;

char ofile[16] = "/dev/null";

char ifile[32];

char buf[32];

if(argc != 2){

printf("usage, %s file, will send contents of file 2 /dev/null\n",argv[0]);

exit(-1);

}

/\* open files \*/

strcpy(ifile, argv[1]);

if((ofd = open(ofile,O\_RDWR)) < 0 ){

printf("error opening %s\n", ofile);

exit(-1);

}

if((ifd = open(ifile, O\_RDONLY)) < 0 ){

printf("error opening %s\n", ifile);

exit(-1);

}

/\* copy from file1 to file2 \*/

read(ifd, buf, sizeof(buf)-1);

write(ofd,buf, sizeof(buf)-1);

printf("copied contents of %s to a safer place... (%s)\n",ifile,ofile);

/\* close 'em \*/

close(ifd);

close(ofd);

exit(1);

}

Pulling the program into gdb with test input aaaaaaa and at a breakpoint after strcpy showed me an esp of 0xffffd690 and the start of the buffer was 0xffffd6e8 and right after the buffer was hexadecimal code for /dev/null, so I assumed I was going to just write the input path (/etc/narnia\_pass/narnia4) and overwrite the output path as well. I briefly considered trying to use a breakpoint after the read and inspecting the bytes but realized gdb probably wouldn’t have permission to the password.

I figured that this program had permission to read /etc/narnia\_pass/narnia4 so I tested that and it worked, but of course it discarded the data.

Then I needed to figure out where to write the output because I couldn't make any files in /narnia. After toying with it for a bit I found in the /README.txt that I could write to /tmp/ so I made a folder and used touch so now I had /tmp/rstevoa/output

I tried to just overwrite the buffer but realized I can't just do that. My first test input was /etc/narnia\_pass/narnia4 + some number of a's to fill out the rest of the 32 byte buffer + /tmp/rstevoa/output and of course that didn't work because (1) it would try to read the whole thing because there was no null byte and (2) there were a bunch of a’s in the argument so it wouldn't work anyway.

I spent half an hour trying to figure out how to make a null byte go into the input:

/etc/narnia\_pass/narnia4\x00 + some number of a's + \x00/tmp/rstevoa/output

...but then I realized that trying to insert null bytes wouldn't work anyway because strcpy would just stop copying as soon as it read saw one.

I spent some time trying to extend the path for the narnia4 file for the input:

///////////////etc/narnia\_pass/narnia4/tmp/rstevoa/output

...but then realized that open also depends on a null byte to stop reading.

I asked a friend of mine and he taught me that links exist. Never would have thought about that! I was about to make a 32 byte link path and realized I'd run into the null byte problem again.

The final setup ended up being something like this:

/tmp/rstevoa/asdfasdfasdfasdfasd <--32 byte long path to directory

/tmp/rstevoa/asdfasdfasdfasdfasd/tmp/rstevoa/io <--full path to input file

^-ifile points here ^-ofile points here

/tmp/rstevoa/io <--full path to output file

ofile and ifile in the program are pointers so they will start reading at the two marked places and I was able to control both of the directories/files. I just needed to make /tmp/rstevoa/asdfasdfasdfasdfasd/tmp/rstevoa/io a link to /etc/narnia\_pass/narnia4. I learned it needed to be symbolic after spending time on Google because I didn’t have permissions for a hard link.

ln -s /etc/narnia\_pass/narnia4 io

(while in the correct directory)

My friend also told me I should be looking into vim so I used that to make the empty file in /tmp/rstevoa

vim io

:wq

cd /narnia

./narnia3 /tmp/rstevoa/asdfasdfasdfasdfasd/tmp/rstevoa/io

And then I was able to cat /tmp/rstevoa/io for the password:

thaenohtai

Note: It also spat out a newline and a bunch of gibberish and I want to know why.