Name: Nhut Trinh

ASU ID: 1215483594

just-execute-me:

Basically just execute the file by command ./just-execute-me

secure-the-house:

First, we need to know the key. In order to obtain the key, I used the fire fighter key and let owner getting the house. Let the owner change the key to new key, then use the new key for attacker to enter the house.

groups:

First, I was searching for the file that has information of all groups. Figured out that it is in etc directory .Use “cd /etc” to change directory. Then “cat group”. Figured out that the group “groups” has a hash of password “aajfMKNH1hTm2”. Using hashcat to break the password by command “hashcat64.exe –a 3 –m 1500 hash.txt –force”. The password is “password”

find-that-pass:

I used scp command to copy the network\_trace file to my local machine. Using Wireshark to read the network\_trace file. Filter out all the junks using “http.request.method == POST”. Only one possible packet shows up. Following its TCP stream and see the encode password = “U2VjdXJpdHlJc1RoZUJlc3Qh”. Using based64 decode found out the password is “SecurityIsTheBest!”

basic-overflow:

Making some researches about buffer overflow, I found the way to solve this challenge by using gdb to know which registers store my input, and which eip register for the function success. The register to run the function is at 0x080484da. By keep printing letter “A”, I notice the buffer is overflowed when it reach 28 letters. I print 28 “A” and concatenate the hex of the register at the end of the string to make it overflow to that instruction register. I use python script to solve this challenge: print ("A"\*28 + "\xda\x84\x04\x08")

read-secret:

I was on the wrong path for this challenge very long. To solve this challenge, I link the two files together, one in my home directory, the other one is the file in read-secret directory. Using command line: ln -s /var/challenge/read-secret/.secret /home/dearCaribou4/.secret. Then I went back to the challenge to run the program ./read-secret then just run l33t.

advanced-overflow:

The same process as the basic-oveflow, but this time the eip address is keep changing every time we run the program. Using gdb to find the eip address of success function which is 0x566215d1. This time I have the program run with the while loop until it matches the address that I found. Using command: while true; do ./advanced-overflow $(python ~/over-flow.py);done. And the script over-flow is: print ("A"\*28 + "\xd1\x15\x62\x56")

search:

I used the same method from read-secret for search. I first linked l33t to my home directory using ln -s /usr/local/bin/l33t test. It links the “l33t” to “test”. The function find in search just returning the result of find. In order to execute l33t, I used –exec, then wrap the whole argument into the quotation. The full command to pass the challenge is: ./search “test –exec {} +”