Homework 3

ECE 568, Spring 2023

This assignment is to be completed in pairs. Directions:

- Do NOT just split the questions up -- both partners should be able to explain every part of the solution to every question.
- Submit written answers to **GradeScope** as a PDF with both partner names/NetIDs.
- GradeScope will prompt you to mark which pages have the answers to which questions. Failure to do so will result in a 5 point penalty to the assignment.
- Write the code for Q1b in a git repo, and provide the URL to that repo on the partner form.
- Fill out the partner form so we can easily credit both group members.

(1) Buffer Overflow [15 points]

These questions revolve around the following paper "Smashing The Stack For Fun And Profit", which you should read carefully:
https://inst.eecs.berkeley.edu/~cs161/fa08/papers/stack_smashing.pdf

(a) [5] Carefully read the full paper "Smashing The Stack For Fun and Profit". Clearly and concisely describe how an attacker can execute a shell using a buffer overflow (or stack smashing) attack.

(b) [10] For this question, you must work with the following provided code:

```
#include <stdlib.h>
#include <stdio.h>

void function(int a, int b, int c) {
    char buffer1[5];
    char buffer2[20];

    ////////////////////////////////////////////////////////////////
    // Fill in your code here. You may only modify this function.
    ////////////////////////////////////////////////////////////////
}

void main() {
    int x;
    x = 0;
    function(1,2,3);
    x = 1;
    printf("x=%d\n", x);
}
```

Your task is to implement code into "void function(int a, int b, int c)" such that when the program is compiled and executed, it will print "x=0". [10]

There are two other requirements:

1. You must compile / run this on your Ubuntu 20 VM.
2. You must use this compile command: gcc -fno-stack-protector -o test test.c

Hint: Refer to the example1.c through example3.c code and descriptions of the "Smashing The Stack for Fun and Profit" paper.

Commit the code for this to a git repo, and submit the repo link in the partner form.

(2) Authentication and Rainbow Tables [20]

a) [5] In Homework 1, you wrote a Django web application. Deploy your application and create a new user with some password of your choosing. What is the hash stored in the back-end database for this user? What algorithm was it hashed with? Does this algorithm meet the password storage requirements described in the course (hashed, salted, with an iteration count)?

b) [5] Assume that an attacker is able to steal a hashed password file from a system. The system uses poor security and does not include a salt value in the hash of each password.

The attacker knows that the system enforces that all passwords are 12 characters in length, and there are 95 valid characters. The attacker has 100,000 GPUs. Each GPU can calculate 100B hashes per second.

How many years (to 1 decimal place) will it take (at most) for the attacker to crack all passwords? Assume 365 days per year.

Max Time = ____ years

c) [5] Assume an attacker wants to prepare a rainbow table for use in cracking hashed passwords. The attacker decides to create a table with 150,000,000 chains, where each chain has a length of 2,000,000.

If each entry of the rainbow table is 16B (8B + 8B), the total size of the rainbow table will be ____ bytes.

d) [5] Assume an attacker has prepared a rainbow table for use in cracking hashed passwords. The rainbow table has 200,000,000 chains and each chain has a length of 1,000,000. The hashed password file has 40 passwords.

If the attacker has a machine that can compute 10 million hashes per second, how long would it take the attacker to crack all passwords (the expected, average time)?

(3) High Availability [10]

a) [5] Assume we have 6 storage drives which are protected with even parity in a striped fashion. The values on each drive are as follows:

```
01100111010 <= Drive 0
10011011100 <= Drive 1
00011011011 <= Drive 2
XXXXXXXXXXX <= Drive 3
10100101101 <= Drive 4
=========================
11011010100 <= Parity
```

A disk failure caused us to lose the data on Drive 3, as shown. The correct data bits of Drive 3 are the following (make sure to just enter 1's and 0's with no spaces/punctuation in between): _____

b) [5] Assume that you are evaluating the "Level of 9's" availability to choose for your server application. You want to make the best cost vs. value tradeoff. Each minute of downtime for your server application costs the company a total of $500. The price per year for the different service levels are: 99.8 = $50,000 99.9 = $150,000 99.99 = $450,000 99.999 = $800,000

Which of the following 4 levels of availability would you choose?

A. 99.8
B. 99.9
C. 99.99
D. 99.999

Justify your answer with math.

(4) Hashing [15]

Assume that you have been given the hash values of a set of secret code words. You must crack each secret code word through a brute force method. The secret code words are of the length indicated and may include both upper and lower case letters and digits (no special characters). The hash used is SHA-256 (SHA-2 algorithm with a 256-bit hash output).

a) [7] Secret code word = 4 characters
Hash = 3803b47609a2a464054659b14a0cdfba92830fb46ee70c03a336d5554b9acad4

b) [8] Secret code word = 6 characters
Hash = 9994a0007e4271061b671424371f3f04dce63520b25ef9036fa45f3439e2f062

c) Secret code word = 10 characters
[This is a stretch goal and worth 5 points of extra credit]
Hash = 6c3d036f677dd8f744b4df9cf517eb2ac93982629e4eb7978336cfcb8b5cedb6

(5) Backups with rsnapshot [20]

NOTE: This question is similar to one posed in my ECE 560 Introduction to Computer Security course. If you have taken that course in a previous semester, you may simply write "<YourName> did this in Computer Security in <semester+year>" for full credit for that student. If your partner did not do it, they're on their own for this question, and they should write "<PartnerName> is doing this alone" above their answer. To be clear, in any case, you may not simply paste in a solution from the other course for this question!

Let's build an automated backup system. Make an additional Linux VM in Duke VCM. This new Linux VM we'll refer to as the **Backup Server**. The thing we'll be backing up is your pre-existing Linux VM; this is the **Backup Client**. You can use this guide, this guide, or any other guide you find with a search for something like "rsnapshot backup linux". Note the following:

- **Backup source**: We're going to back up your user home directory on the backup client machine. This is `/home/<NETID>`.
- **Backup destination**: As root, make a directory `/backup`.
- **SSH keys needed**: Because the source and destination are different machines, you'll need to set up SSH keys so that the backup server can *pull* data as needed.
- **Frequency**: Set up nightly and weekly intervals.
- **Retention**: You should retain seven nightly backups and four weekly backups.
- **Automation:** You do NOT need to do cron-based automation. Once you have rsnapshot configured, you can just run "rsnapshot nightly" to perform the backup that would happen nightly, and "rsnapshot weekly" to perform the backup that would happen weekly. You can run these commands as often as you wish; there's nothing in the software that actually needs the backups to be done nightly/weekly as opposed to every few dozen seconds (i.e., rsnapshot does not actually care that they're called "nightly" or "weekly").

Provide screenshots or a terminal log of these steps in your writeup. [5]

Provide the relevant portions of rsnapshot.conf. [3]

Let's test it. Open one terminal window as root, and another terminal window as the non-root user. Do the following, and **for each step, show the process via screenshots or terminal logs. Label or otherwise identify each step you're doing in your screenshot/log.**

1. As the user on the backup client, add some content to the user home directory. [1]
2. As root on the backup server, do several nightly and weekly backups to populate your backups. [1]
3. As the user on the backup client, "corrupt" an important file (overwrite or delete it). [1]
4. As root on the backup server, take another few nightly backups to simulate time passing. [1]
5. Copy the appropriate backup from the backup server to the client's home directory, thus restoring the damaged file. Show that it is restored. [3]

Note: If your SSH MFA is interfering with this, you may either make a separate user without MFA enabled, or disable MFA altogether.

Finally, describe a scenario in which you'd need an older weekly backup as opposed to just the most recent nightly backup. [5]

(6) Backdoors [20]

In this question, we will act as attackers and create a privilege escalation backdoor. Then we'll switch to acting as defenders and locate/disable this backdoor. Because we will be making the system less secure on purpose, do the following question on a totally separate scratch Ubuntu Linux VM from Duke VCM which you can delete when done.

First, we'll act as attackers. We'll assume that we've already gained root access to this victim machine, and using this root access, create a **privilege escalation backdoor** so that an unprivileged user (i.e., someone without sudo access) can become root (UID 0) in the future. You have a variety of approaches to do this:

- Creation or modification of a setuid root binary
- Changes to the passwd and/or shadow files
- Changes to the sudo configuration
  (must be sneaky and non-obvious -- just making a rule that clearly gives sudo access doesn't count)
- Addition of a root-privileged daemon
- Creation of a malicious kernel module
- Many more

This backdoor may be pre-auth (allowing root privilege even without logging into the machine) or post-auth (requiring a login to an unprivileged, non-sudo-capable account). If making a post-auth backdoor, create an unprivileged account to demonstrate it working.

Document how you did the above in detail, and document proof of it working. [10]

Now we will switch our role to that of the defender. Without cheating and using knowledge of the specifics of your backdoor, document the set of commands a system administrator could use to detect a backdoor such as yours. For example, if you make a setuid root binary, how could an administrator scan for such things and find yours? If you made a root-privileged daemon, how could you find it running?

Document your search procedure, and once you use it to "find" your backdoor, document how to remove or disable it. [10]

Note: Up to +5 points extra credit is available for especially novel or stealthy backdoor implementations.